Poky meets Debian: Understanding How to Make an Embedded Linux by Using an Existing Distribution's Source Code

Yoshitake Kobayashi, TOSHIBA

Embedded Linux Conference 2015

Information

- **All source code are available at the following URL:**
  - https://github.com/ystk
  - Poky-debian
    - https://github.com/ystk/poky-debian
    - Branch: master
  - Meta-debian
    - https://github.com/ystk/meta-debian
    - Branch: daisy

Can we create a “Standard Embedded Linux Distribution”?

Embedded Linux Conference Europe 2014

It is really difficult!

How to choose a suitable distribution for you?

- **There are a lot of Linux distributions**

- **Need to be considered the following**
  - **Usecases**
  - Supported CPU architectures
    - X86, ARM, PowerPC, etc.
  - Number of packages
    - How many packages are enough?
  - Security fix support period for some distributions
    - Centos 10 years
    - OpenSUSE 3 years
    - Debian 3 years+2years(LTS)
    - Ubuntu 5 years
    - Gentoo Incremental update (rolling release)

Why Debian?

- **Would like to use Desktop and Embedded with a same source code base**
- **Would like to build embedded Linux environment for the following CPUs**
  - X86(32bit and 64bit)
  - ARM
  - PowerPC
  - might be others
- **Would like to use same package version in one major release**

Why not Emdebian?

- **http://www.emdebian.org/**
  - As of July 2014, updates to the Emdebian distributions ceased. There will be no further updates and no further stable releases.

- **Would like to customize more than Emdebian's way**

Why Poky?

- **Poky is one of the most popular reference distribution for embedded Linux**
- **Would like to share the knowledge**
  - Bitbake
  - Recipes
  - Tools

What we want?

- **Use Debian's source code**
- **Make custom embedded Linux environments**
- **Would like to change it to open**
- **Share knowledge with Yocto Project**

Scope of this presentation

- **Introducing of the following two implementation**
  - Poky-debian (BAD manners)
  - Meta-debian (GOOD manners)

- **Out of scope**
  - What is the Yocto Project

Compareing Poky-debian and Meta-debian

	Poky-debian	Meta-debian
Poky version	Edison	Daisy
Debian version	6 (Squeeze)	8 (Jessie)
Kernel	LTSI + RT patch	LTSI + RT patch
Distribution Support	Debian 6	Debian 8

poky-debian

Introduction of Poky-debian

- **Replacing meta layer to adapt Debian source**

- **Source code**
  - https://github.com/ystk/poky-debian.git

What is "poky-debian" ?

- **Based on**
  - Poky (Edison)
  - Debian 6 (squeeze-lts)

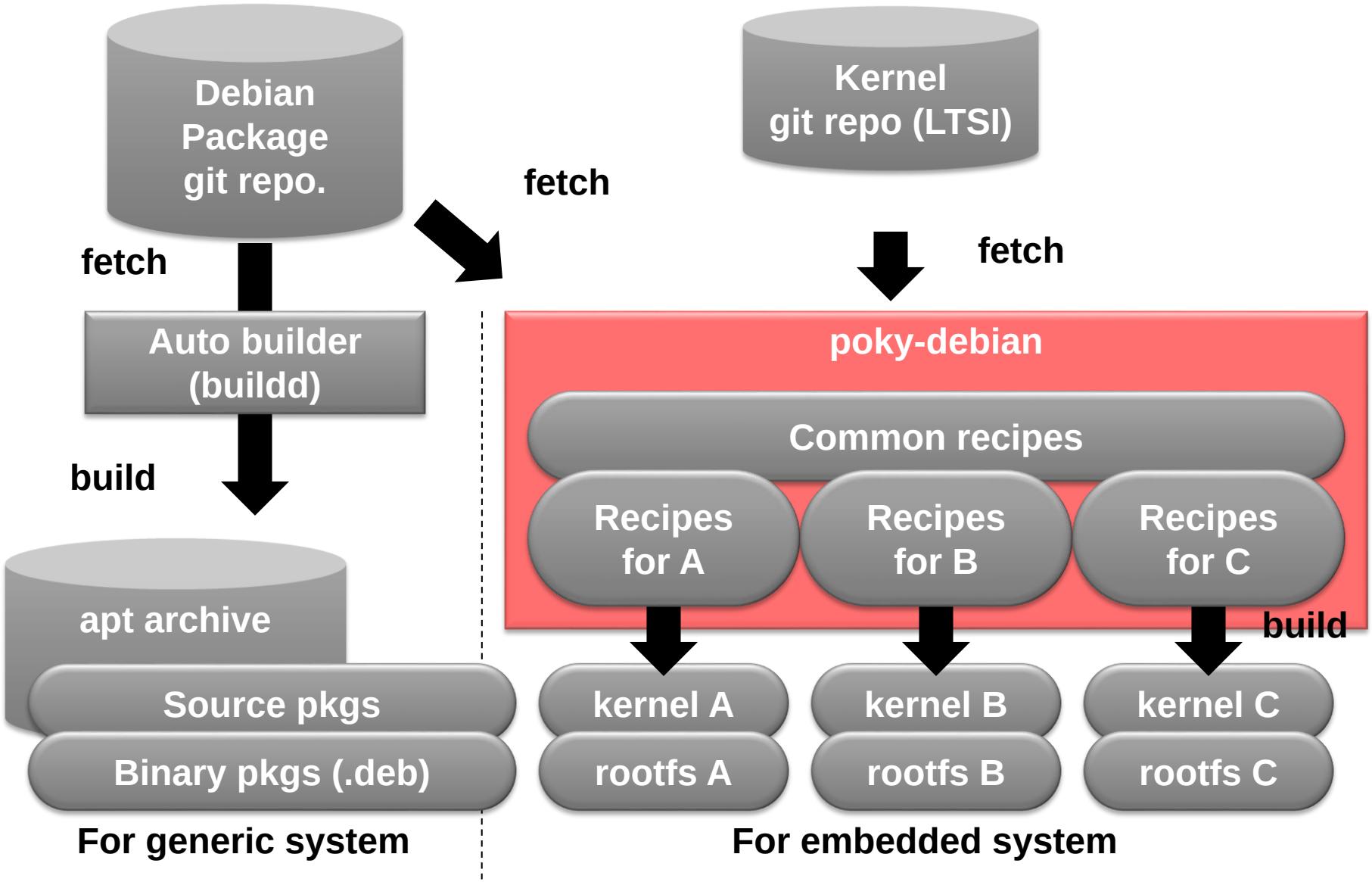
- **Generate Linux environment for embedded systems (x86, ARM, PowerPC, etc.)**
  - Based on "poky" developed by Yocto Project
  - Kernel, rootfs and toolchain only from Debian source codes
  - Common sources + board-specific customization

- **Support distro**
  - Debian 6 LTS (squeeze-lts) only
  - All build test has been done on x86-32bit environment

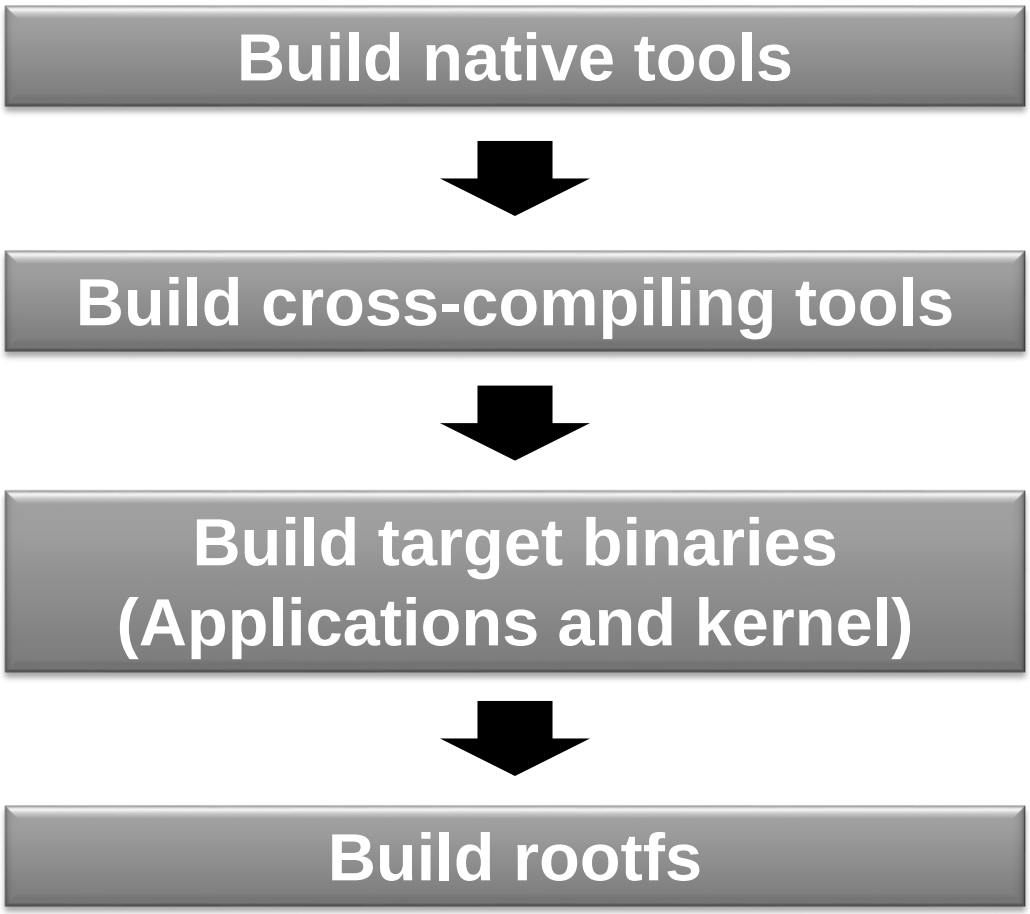
Goal of poky-debian

- **Build Debian based Linux environment for many embedded systems as quickly as possible**
- **Generate "minimal" environemt for each requirement**
  - Ex: Consists only busybox and libc
- **Share problems between all systems generated by it**
- **Share know-how for embedded Linux system implementation**
- **Improve traceability of all sources & customization**

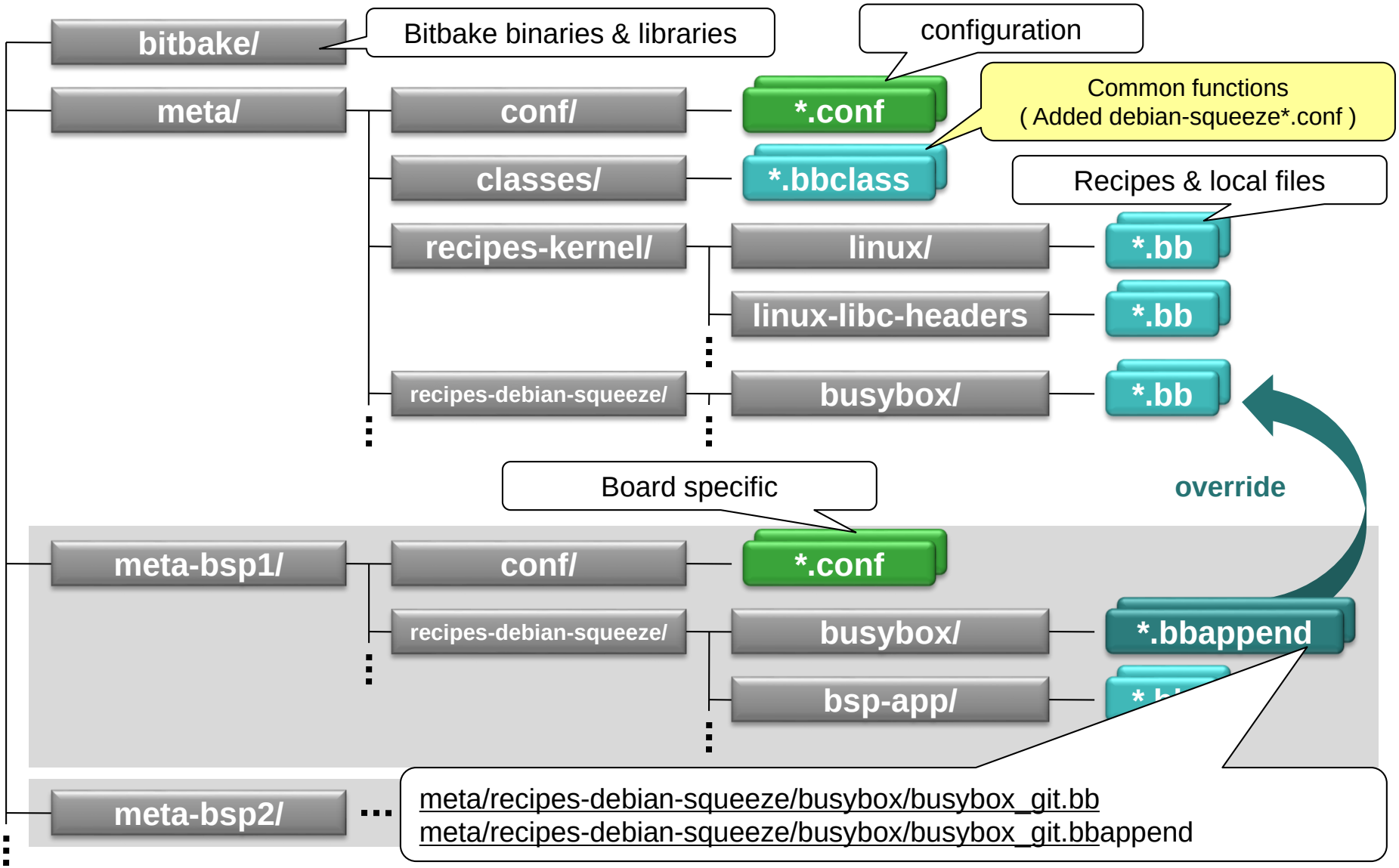
Packaging system structure



Build flow


Build native tools
Build cross-compiling tools
Build target binaries
(Applications and kernel)
Build rootfs

Directory structure of poky-debian



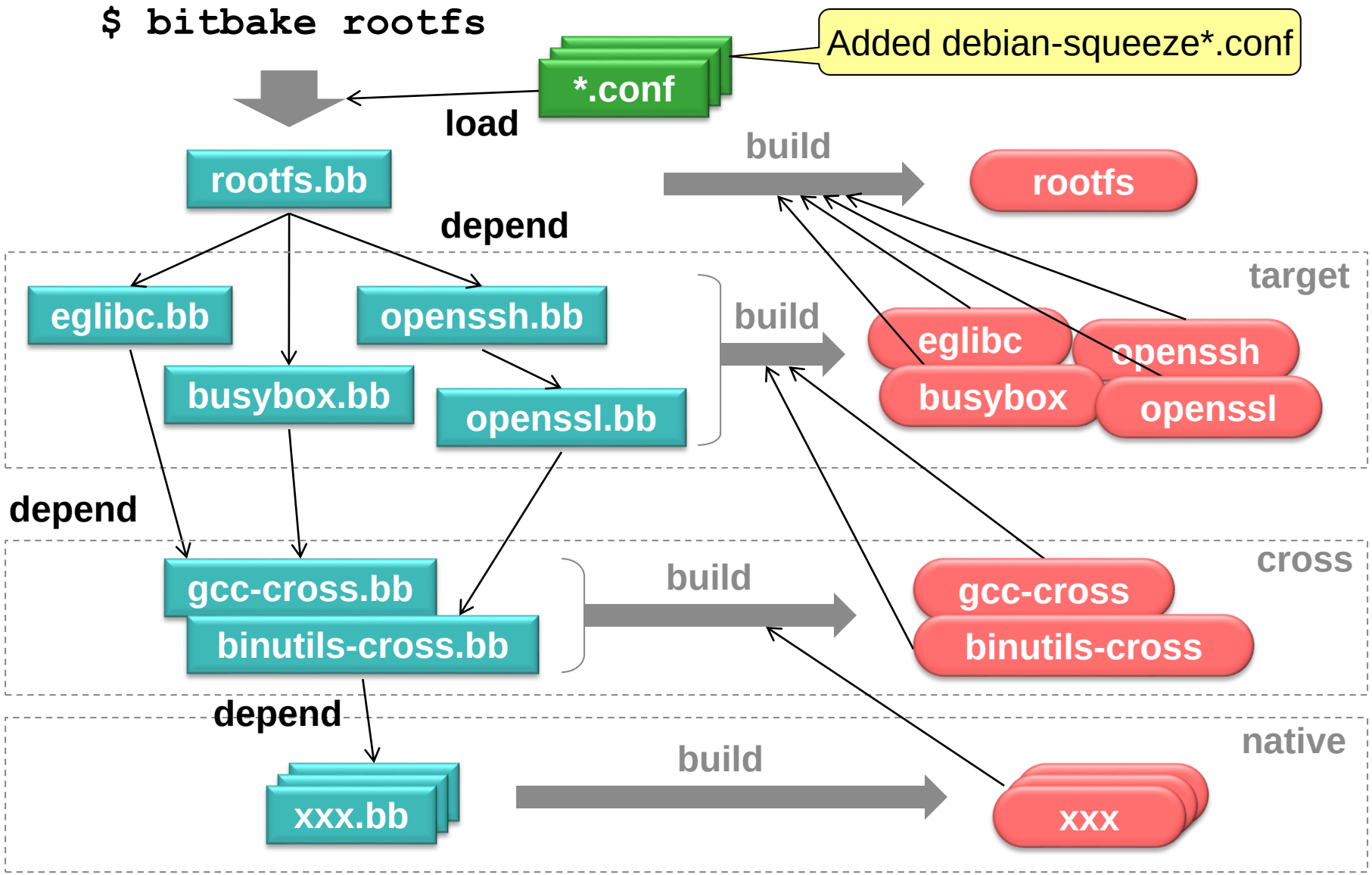
Core components

- **bitbake**
  - Core build tool of poky
  - Parse all recipes and run all build actions
  - Ex: To build busybox, run `$ bitbake busybox`

- **Recipes (*.bb)**
  - Defines how to build each package, kernel, rootfs, toolchain, etc.
  - Written in shell & python based script
  - Actions in some recipes depend on other recipes

- **Configuration (.conf)**
  - Defines machine or build environment-specific values

The relation of core components



Build tasks

- **Each recipe has the following "task":**
  1. do_fetch*: Fetch source to the download directory
  2. do_unpack*: Checkout git, unpack archives, etc.
  3. do_patch*: Apply local patches
  4. do_configure: Do `$ ./configure`
  5. do_compile: Do `$ make`
  6. do_install: Do `$ make install`

  ……..

- **Each task is defined as a function in a recipe**

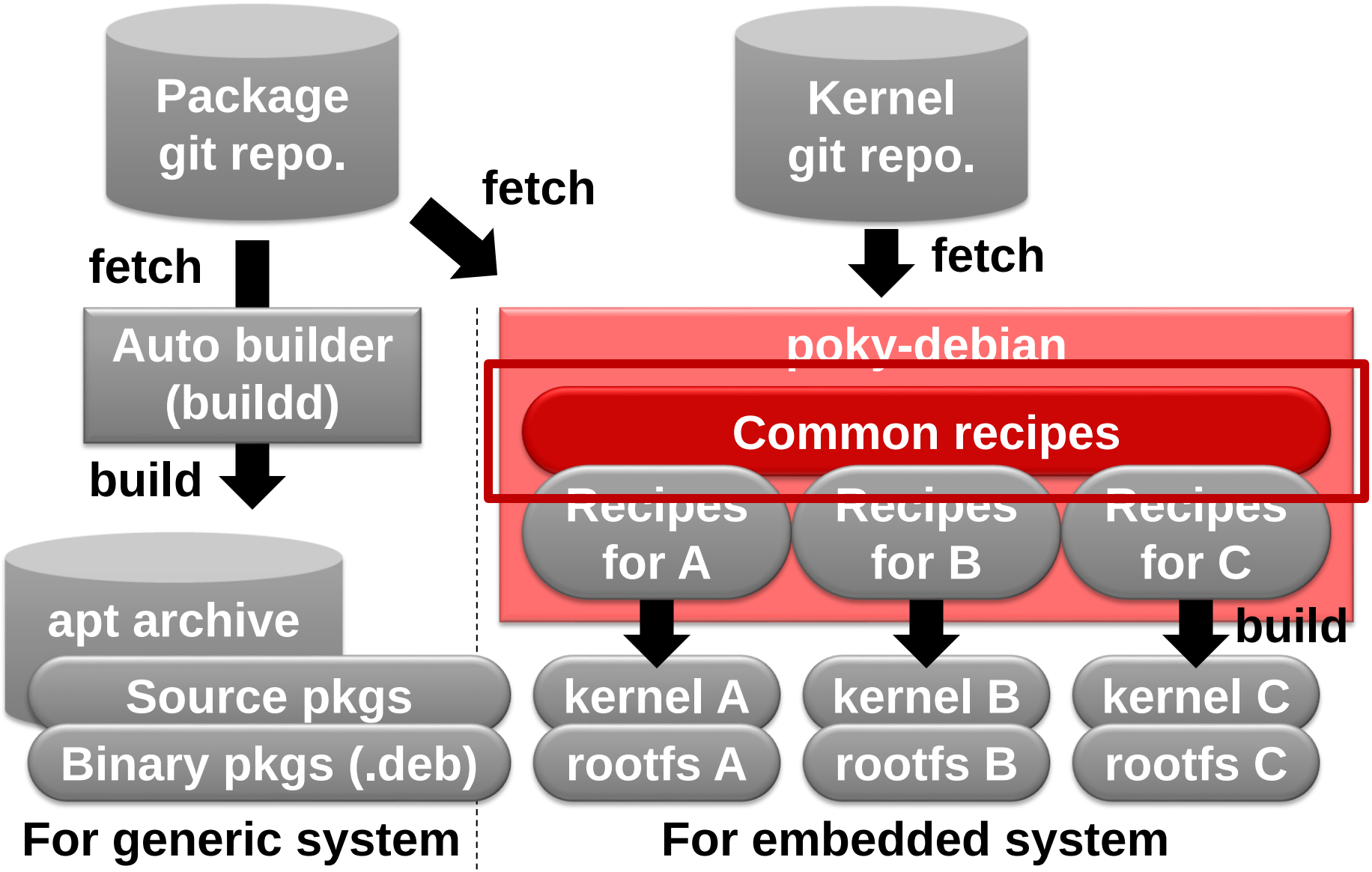
Developing recipes for poky-debian

Issues

- **At least 200-300 packages are required to apply poky-debian to embedded systems**
  - Social Infrastructure, power system, train/highway, medical, …

- **Need to implement & update many "common" recipes for Poky-debian**
  - "common" means "not depend on target system"
  - All recipes for board X are created based on "common" recipes

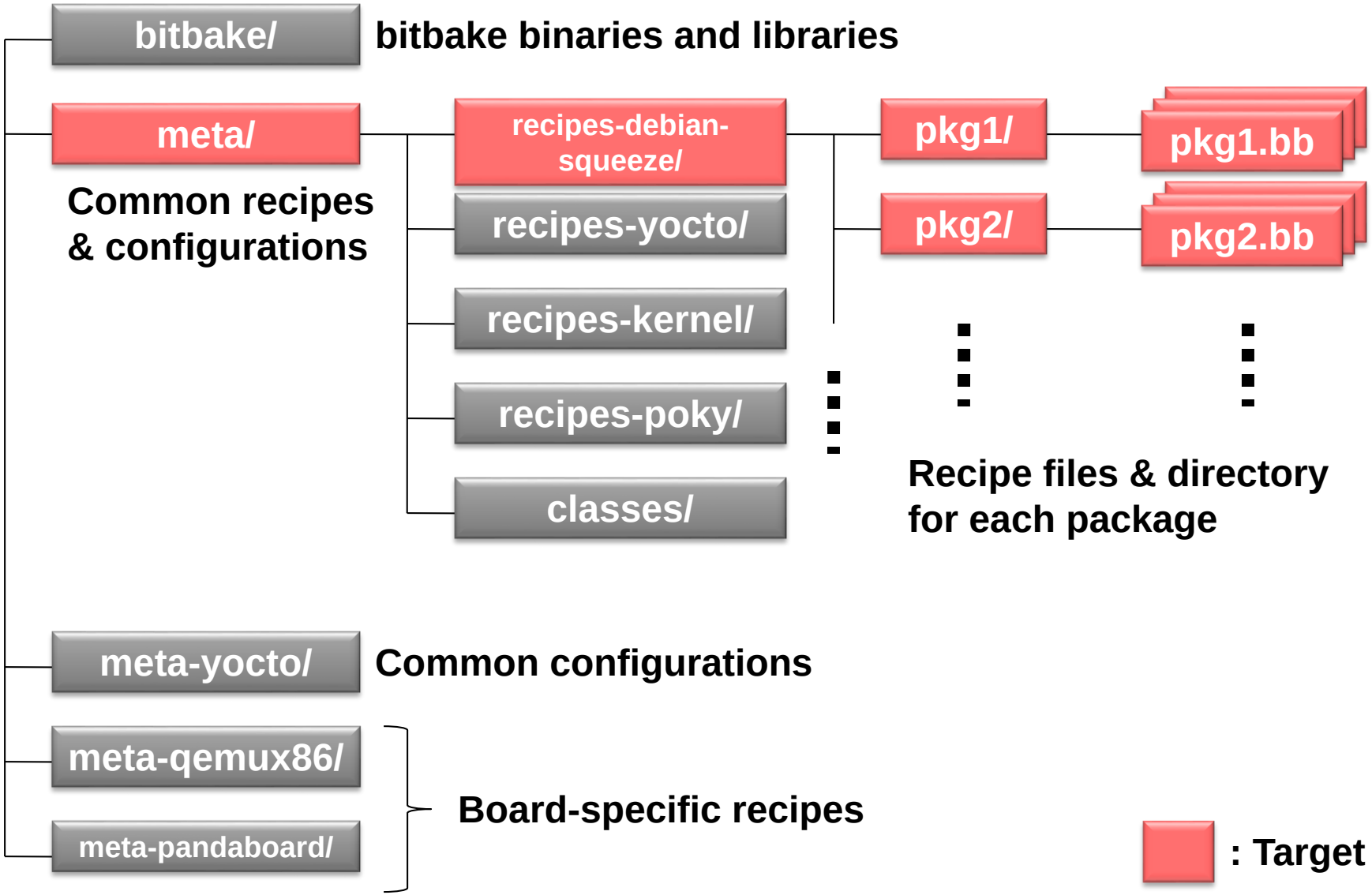
Packaging system structure (target)



Why BAD manners?

- **First step**
  - Move all original recipes (recipes-core) to somewhere… ☹

Directory structure



How to implement a recipe for "xyz" (create)

- **Implement a recipe for "xyz" from scratch if there is no original poky recipe**

- **Need to check the following:**
  - Proper configure & make options
  - Build dependency
  - Run-time dependency
  - How to split output binaries to packages (.debs)
  - And more…

Example: Recipe for "hello"

- Implement a recipe to build the following program

```
#include <stdio.h>

int main()
{
        printf("hello¥n");
        return 0;
}
```

- Makefile

```
default: clean hello
hello: hello.o
clean:
        rm -f hello *.o
```

Example: Recipe for "hello"

```
LICENSE = "tmp"
LIC_FILES_CHKSUM = ¥
"file://COPYING;md5=d41d8cd98f00b204e9800998ecf8427e"

SRC_URI = "file://src"

S = ${WORKDIR}/src
B = ${S}

do_install() {
        install -d ${D}/${bindir}
        install -m 0755 ${B}/hello ${D}/${bindir}
}
```

How to implement a recipe for "xyz" (modify)

1. **Copy recipes from original poky directory**
   - `$ cp -r meta/ORIGINAL/recipes-*/xyz meta/recipes-debian-squeeze`

2. **Modify recipe files**
   - Change to fetch all sources from git repository
     (Original recipes fetch sources from upstream site or Yocto Project source repository)

3. **Build test => Usually error occurs :(**
   - Because of lack of source repositories, patch rejects, missing source files, build-dependency, include / link problems, ···...

4. **Re-modify recipe files to fix errors**

Example: Recipe for "sed"

```
#
# sed_4.2.1.bb
#

DESCRIPTION = "sed is a Stream EDitor."
HOMEPAGE = "http://www.gnu.org/software/sed/"
...

#SRC_URI = "${GNU_MIRROR}/sed/sed-${PV}.tar.gz"

#SRC_URI[md5sum] = "f0fd4d7da574d4707e442285fd2d3b86"
#SRC_URI[sha256sum] =
"8773541ce097fdc4c5b9e7da12a82dffbb30cd91f7bc169f52f05f93b7fc3060"

inherit autotools update-alternatives gettext

...

BBCLASSEXTEND = "native"

#
# debian
#

inherit debian-squeeze
```

Poky-debian Quick Start

Setup poky-debian source tree

- **Disable dash on Debian**

```
$ sudo dpkg-reconfigure dash
```

  - Select "no" in menu

- **Download poky-debian**

```
$ mkdir $WORKDIR; cd $WORKDIR
$ git clone git://github.com/ystk/poky-debian.git
$ cd poky-debian
```

- **Install dependent packages**

```
$ ./scripts/install-deps.sh
```

Setup poky-debian (Need to do every time)

- **Setup build environment for target board**

```
$ . ./poky-debian/setup.sh qemux86
```

  - This command means that we use "qemu for x86" as the target board

- **Build**

```
$ bitbake core-image-base
```

  - Output directory: build-qemux86/tmp/deploy/

Meta-debian

Lessons learned from Poky-debian development

- **Poky-debian has a lot of local rules**
- **As the result**
  - poky-debian cannot follow the Poky's development tree
  - Difficult to make it open

- **Next time, we would like to create more friendly one with Poky**

Meta-debian

What is meta-debian?

- **Extending poky recipes to use Debian sources**
- **Based on newer Poky provided by Yocto Project**
- **Provide "meta-debian“ only**
  - All recipes and configuration are included in it
  - Completely separated from OE-core (meta) and other layers
- **Source code**
  - https://github.com/ystk/meta-debian.git

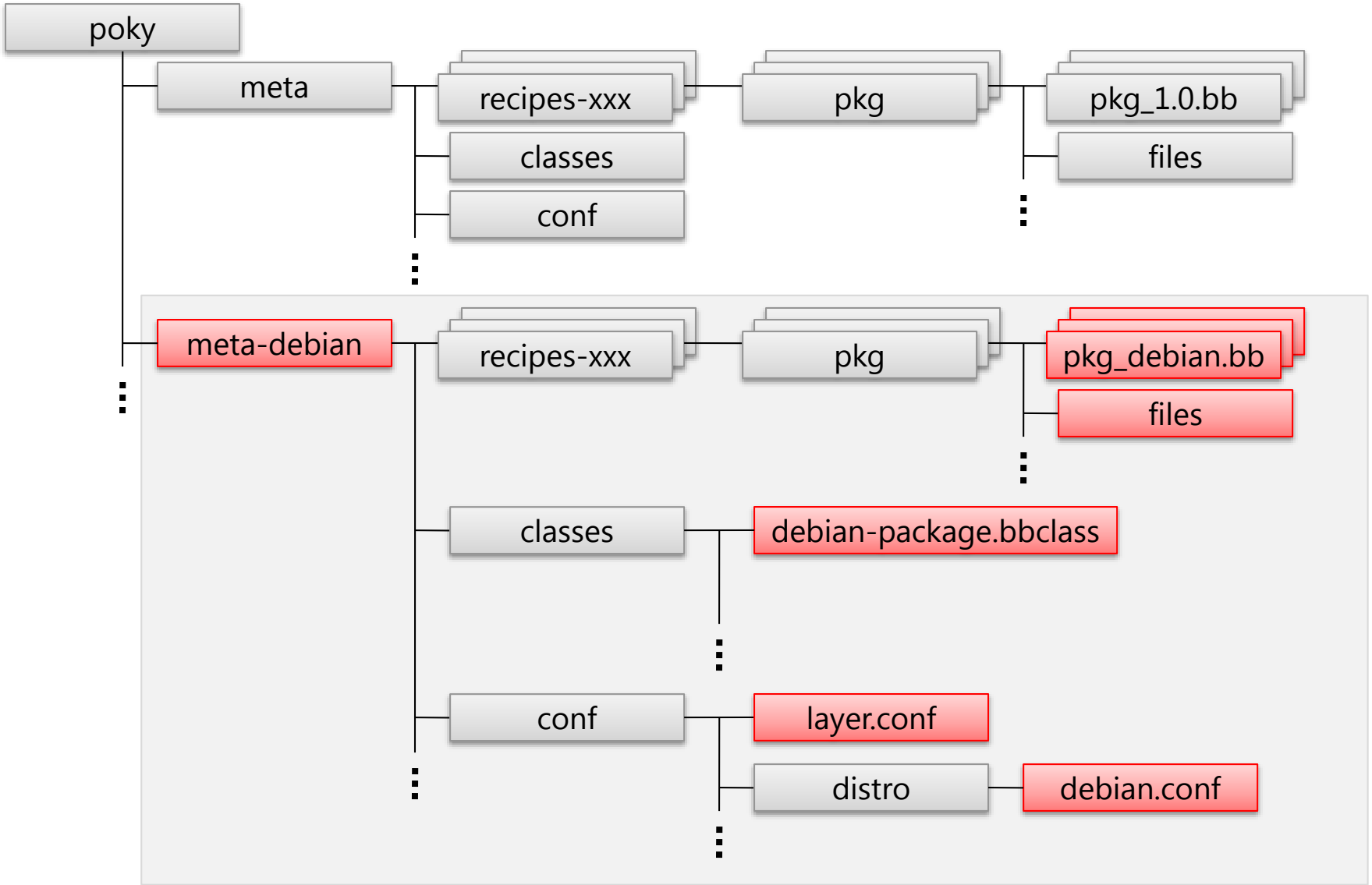
Purpose

- **Make easy to use Debian source through Poky**
- **Would like to contribute something to Debian long term support and use the source code with Poky**
- **Keep reproducibility of each build**
- **Output more detail information about package and license**

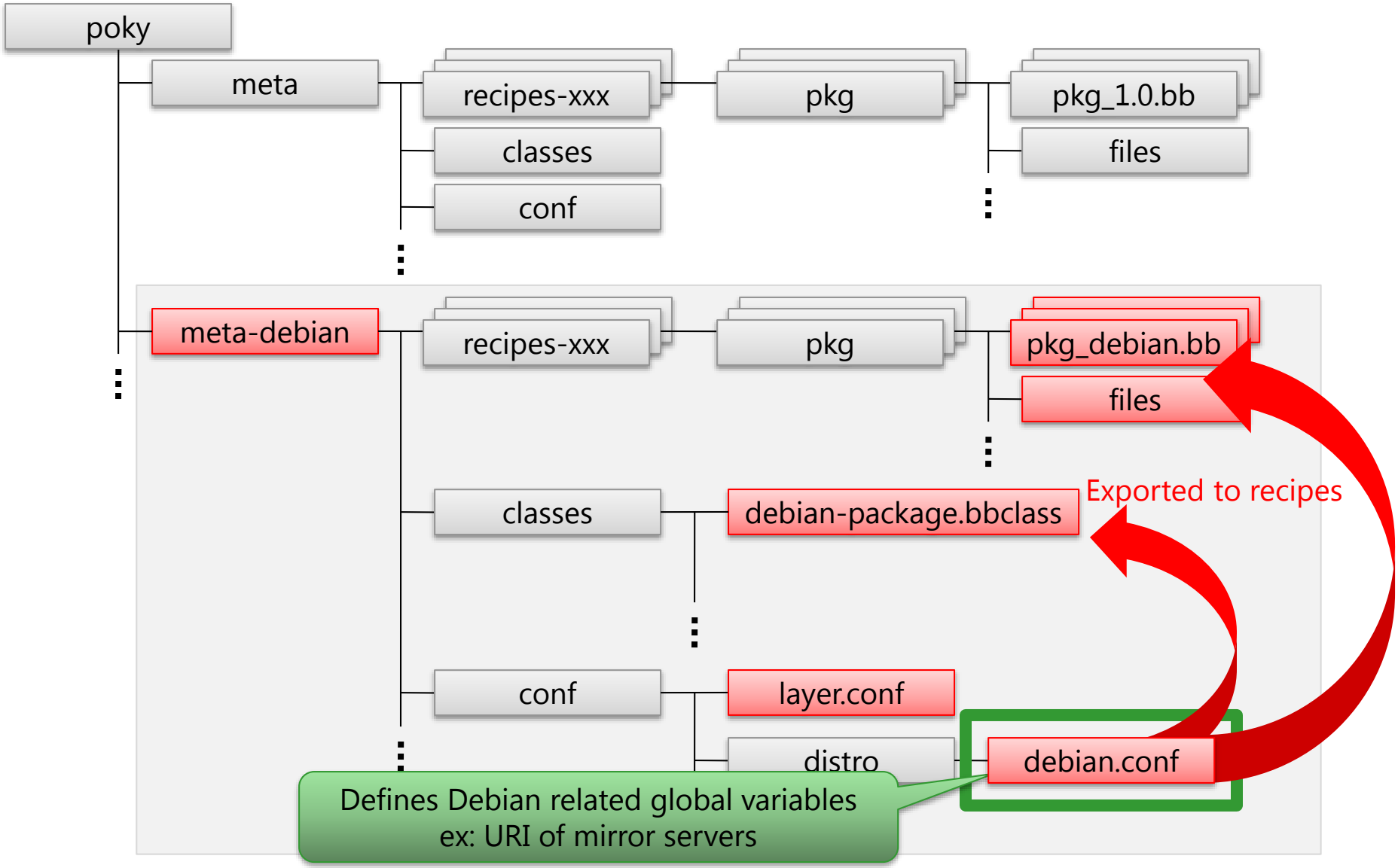
meta-debian : Basic information

- **Debian version**
  - Debian GNU/Linux 8 (jessie)
- **Poky version**
  - Yocto Project 1.6 Daisy

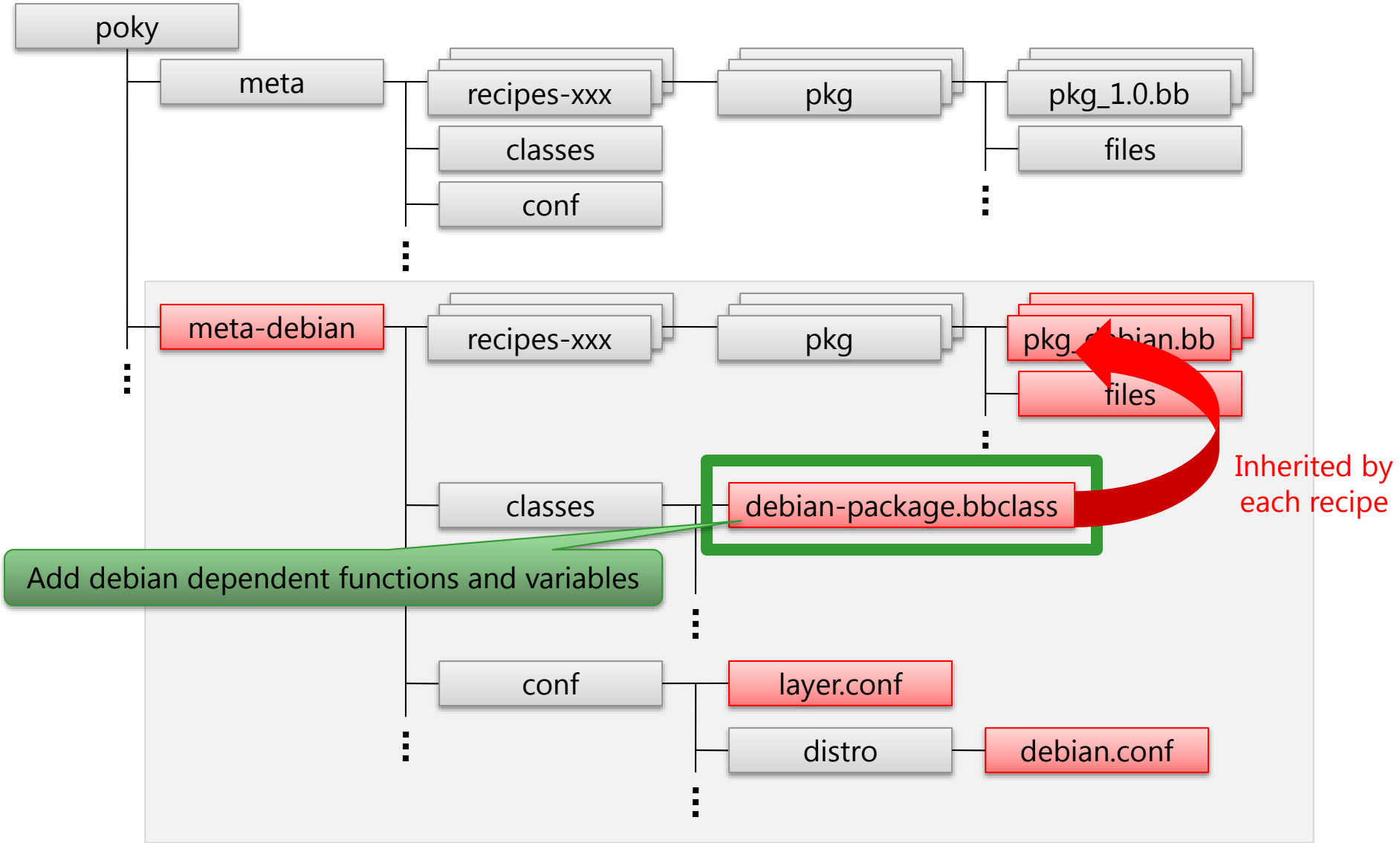
Directory structure of poky and meta-debian



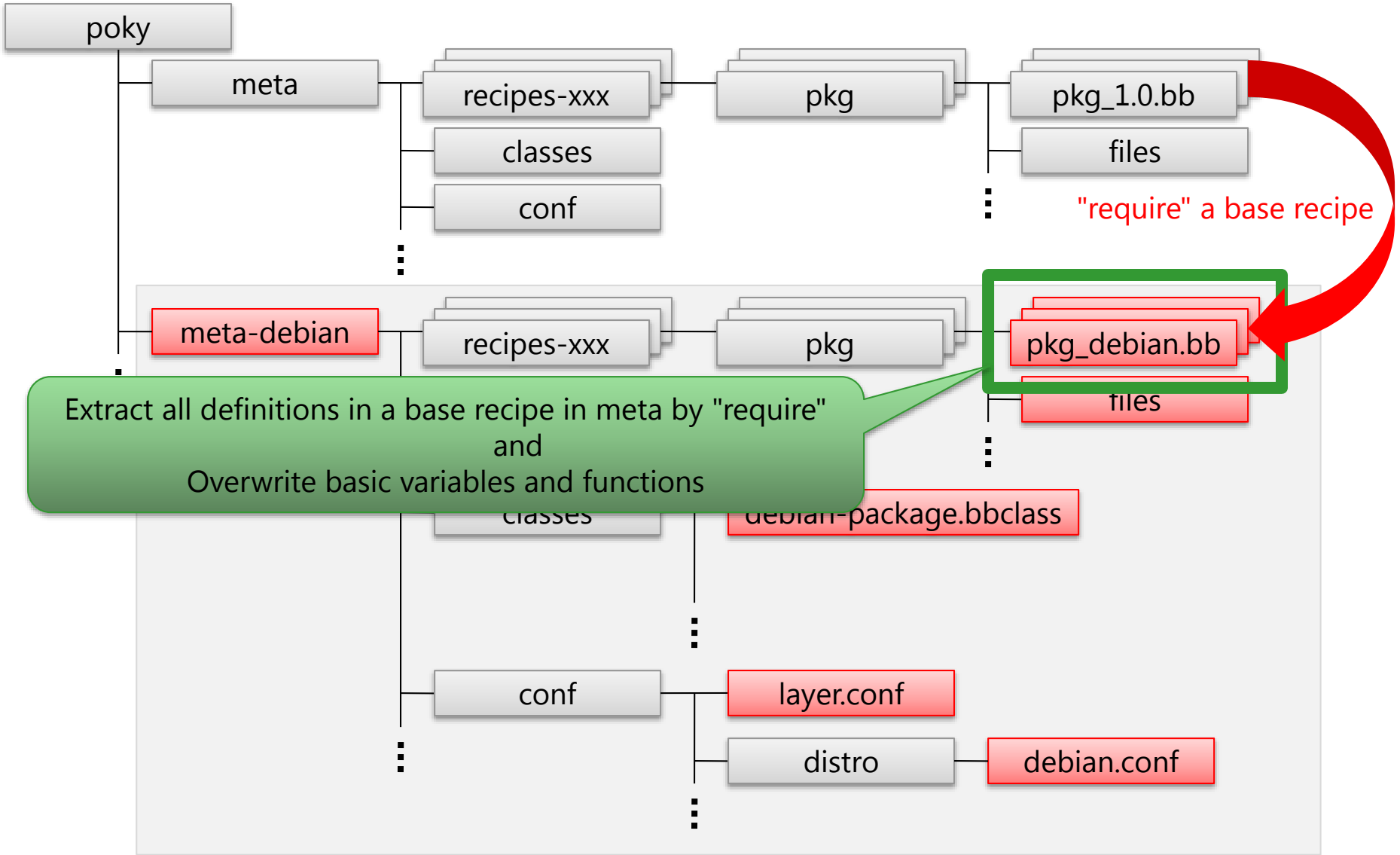
Directory structure of meta-debian



Directory structure of meta-debian



Directory structure of meta-debian



Creating a recipe

Sample recipe: quilt-native

- **meta-debian/recipe-debian/quilt/quilt-native_debian.bb**

```
require recipes-devtools/quilt/${PN}_0.61.bb
FILESEXTRAPATHS_prepend = "${COREBASE}/meta/recipes-devtools/quilt/quilt:"

inherit debian-package

DPR = "0"

LICENSE = "GPLv2"
LIC_FILES_CHKSUM = "file://COPYING;md5=94d55d512a9ba36caa9b7df079bae19f"

SRC_URI = " ¥
file://install.patch ¥
file://run-ptest ¥
file://Makefile ¥
"

debian_patch_quilt() {
        ...
}
```

Step 1: Create files and directories

- **Create directory meta-debian/recipe-debian/quilt**
  - Same path as the base recipe in poky/meta
- **Touch ${PN}_debian.bb**
  - PN = quilt-native
  - PV = debian
- **Create sub directories including local files if needed**
  - Ex: files, ${PN}, ${BPN}, etc.

Step 2: Define a base recipe

```
require recipes-devtools/quilt/${PN}_0.61.bb
FILESEXTRAPATHS_prepend = "${COREBASE}/meta/recipes-devtools/quilt/quilt:"
```

- **Require a base recipe included in meta**
  - Find a base recipe from poky's tree
  - Sometimes ${PN} is not the same name as Debian source package
    - Ex: libusb
      - PN = "libusb1"
      - Debian source package name: "libusb-1.0"
- **Add directories to FILESEXTRAPATHS if needed**
  - By default, bitbake doesn't search files from the directory that includes the base recipe which requires

Step 3: Inherit debian-package

```
require recipes-devtools/quilt/${PN}_0.61.bb
FILESEXTRAPATHS_prepend = "${COREBASE}/meta/recipes-devtools/quilt/quilt:"

inherit debian-package
```

- **Added/overwritten variables**
  - PN = quilt-native
  - PV = git${SRCPV}
  - PR = r0 (Same as PR in the base recipe)
  - DPN ?= quilt (${BPN})
  - DEBIAN_UNPACK_DIR ?= "${WORKDIR}/git"
  - S = "${DEBIAN_UNPACK_DIR}"
  - etc.
- **Exported functions**
  - do_debian_patch

Step 4: Add PR

```
require recipes-devtools/quilt/${PN}_0.61.bb
FILESEXTRAPATHS_prepend = "${COREBASE}/meta/recipes-devtools/quilt/quilt:"

inherit debian-package

DPR = "0"
```

- **"0": Initial version**
- **Don’t forget to add 1 to this value when you modified**
  - 0 -> 1 -> 2 …

Step 5: Add license information

```
require recipes-devtools/quilt/${PN}_0.61.bb
FILESEXTRAPATHS_prepend = "${COREBASE}/meta/recipes-devtools/quilt/quilt:"

inherit debian-package

DPR = "0"

LICENSE = "GPLv2"
LIC_FILES_CHKSUM = "file://COPYING;md5=94d55d512a9ba36caa9b7df079bae19f"
```

- **Always investigate source tree and set correct values**
  - Don’t copy them from the base recipe
  - Usually license information can be found in COPYING*, LICENSE*
- **Choose a license name from meta/files/common-licenses if exists**

Step 6: Overwrite SRC_URI

```
require recipes-devtools/quilt/${PN}_0.61.bb
FILESEXTRAPATHS_prepend = "${COREBASE}/meta/recipes-devtools/quilt/quilt:"

inherit debian-package

DPR = "0"

LICENSE = "GPLv2"
LIC_FILES_CHKSUM = "file://COPYING;md5=94d55d512a9ba36caa9b7df079bae19f"

SRC_URI = " ¥
file://install.patch ¥
file://run-ptest ¥
file://Makefile ¥
"
```

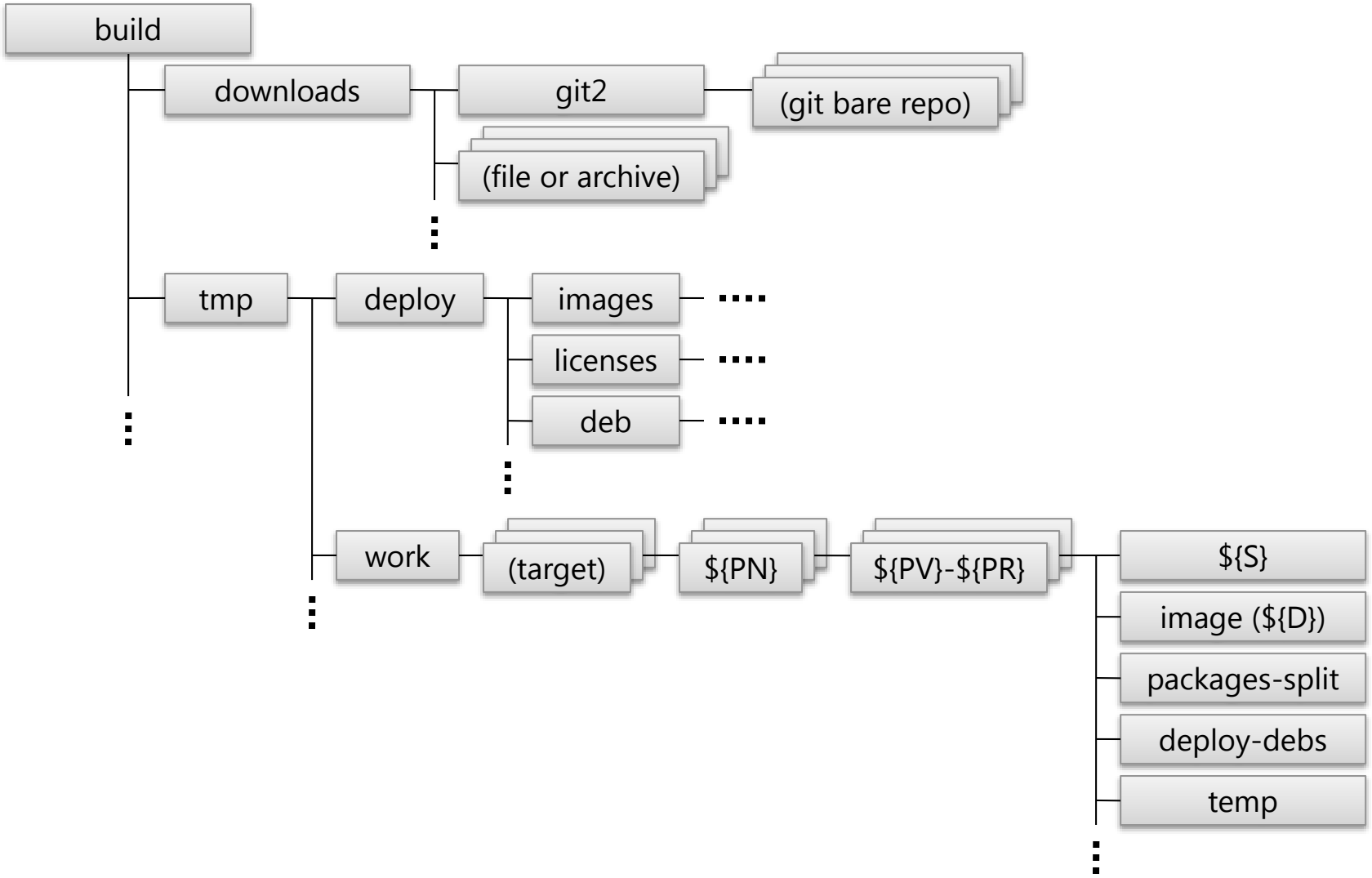
- **Exclude the upstream source code**
  - Ex: http://download.savannah.gnu.org/releases/quilt/quilt-${PV}.tar.gz
- **Add all local files (scripts, patches, etc.) to SRC_URI**
  - Need to solve problems if theses files conflict with Debian source

Other rules about modifying recipes

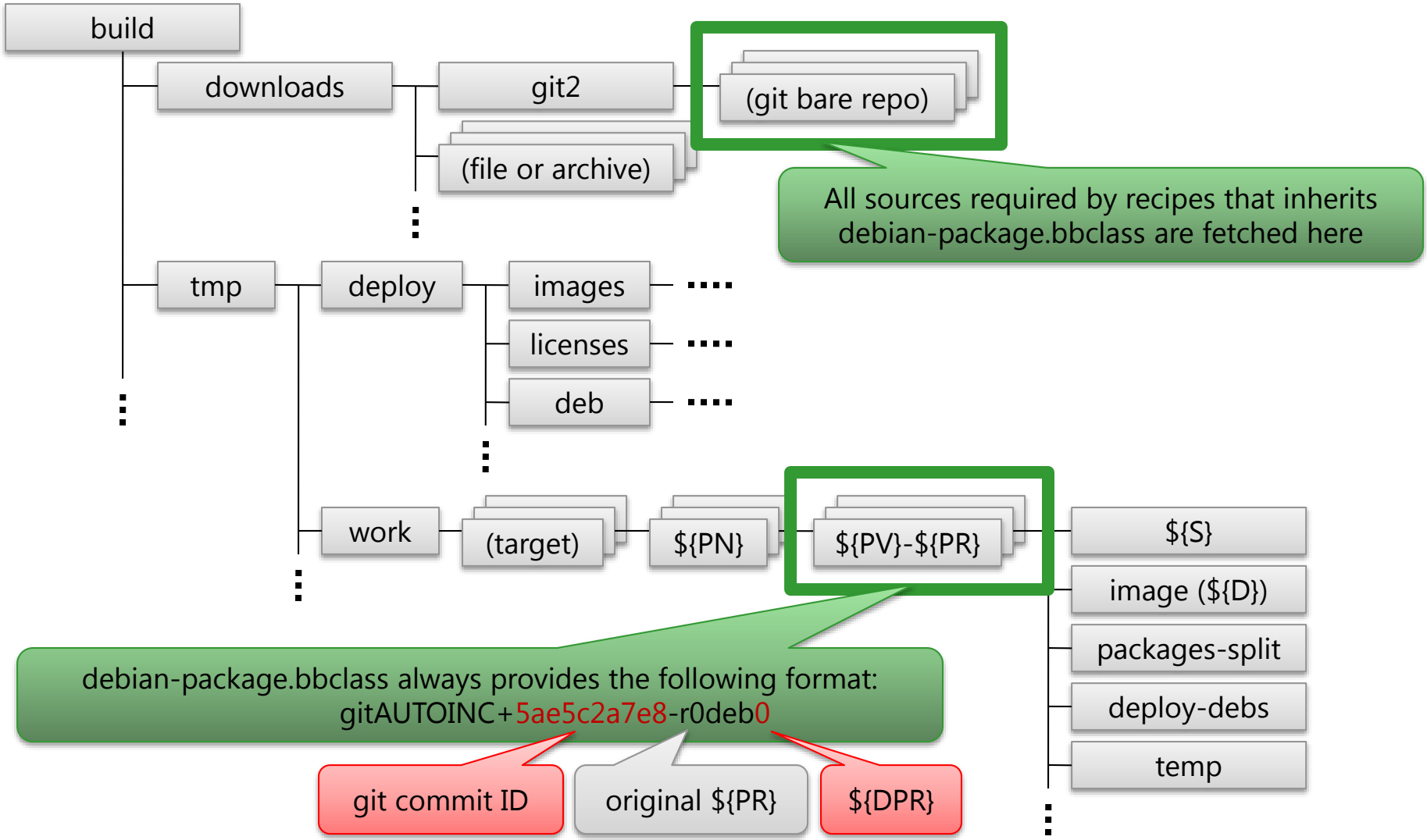
- **Please don’t modify Poky. All modification has to be done to meta-debian**
- **Don’t include specific hardware, company or project related name and functions into meta-debian**
- **Add only really essential DEPENDS and RDEPENDS**
- **Always leave comments**
  - **Why** did you modify so? Nobody knows, you will forget it ☹

Build

Build directory structure



Build directory structure



Build flow

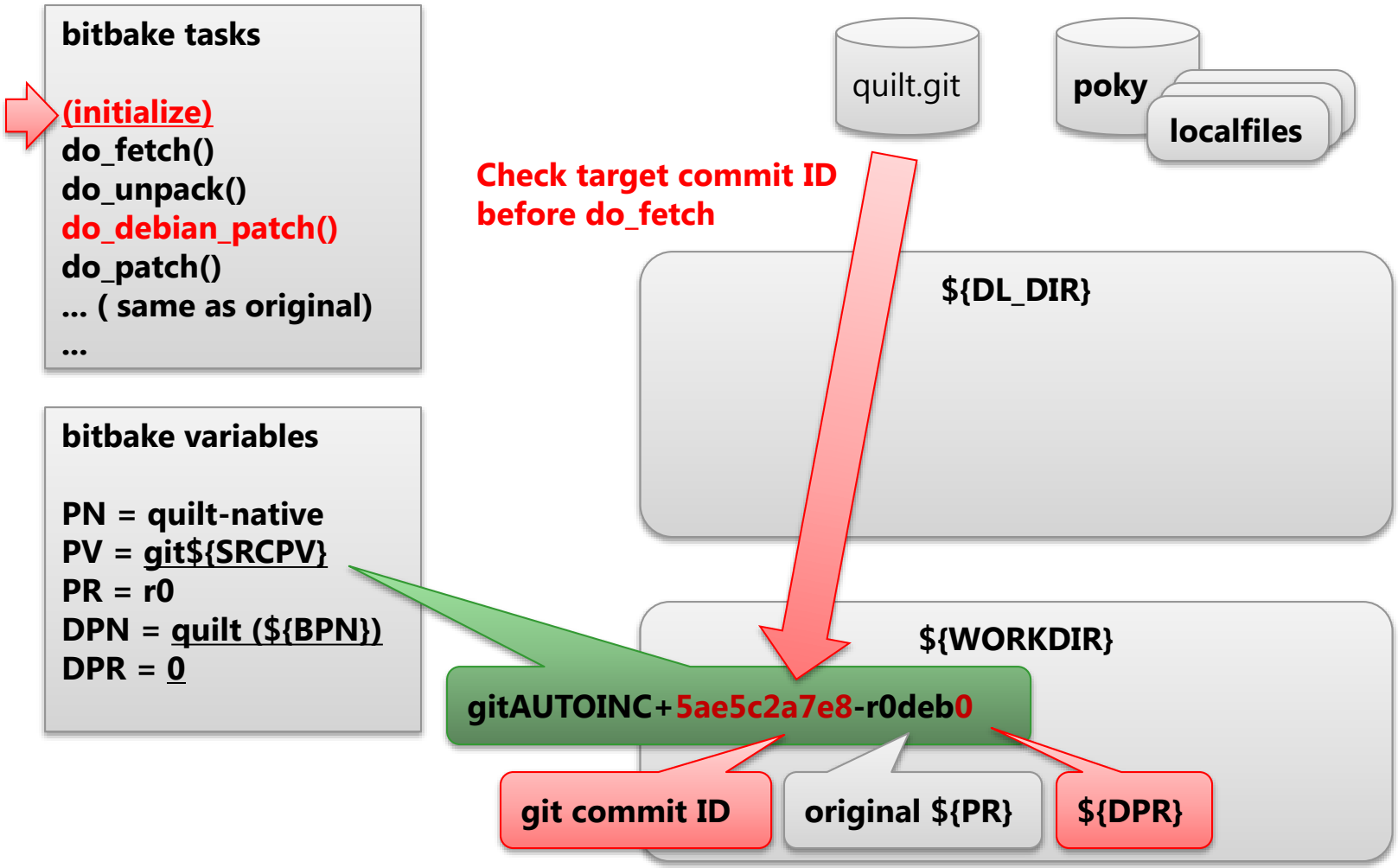
bitbake tasks

(initialize)
do_fetch()
do_unpack()
do_debian_patch()
do_patch()
... (same as original)
...

bitbake variables

PN = quilt-native
PV = debian
PR = r0
DPN = N/A
DPR = N/A

Build flow


bitbake tasks
(initialize)
do_fetch()
do_unpack()
do_debian_patch()
do_patch()
... (same as original)
...
bitbake variables
PN = quilt-native
PV = git${SRCPV}
PR = r0
DPN = quilt (${BPN})
DPR = 0
quilt.git
poky
localfiles
Check target commit ID
before do_fetch
${DL_DIR}
${WORKDIR}
gitAUTOINC+5ae5c2a7e8-r0deb0
git commit ID
original ${PR}
${DPR}

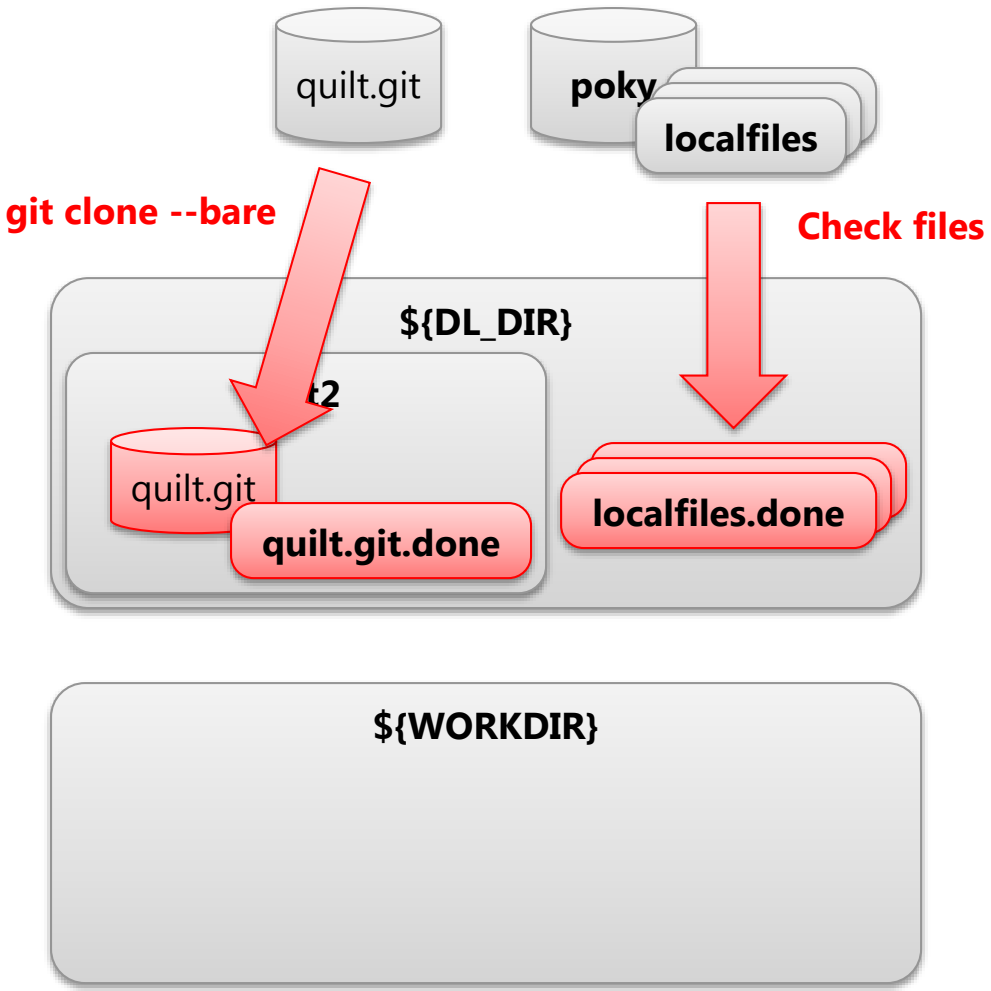
Build flow

bitbake tasks

(initialize)
do_fetch()
do_unpack()
do_debian_patch()
do_patch()
... (same as original)
...

bitbake variables

PN = quilt-native
PV = git${SRCPV}
PR = r0
DPN = quilt (${BPN})
DPR = 0



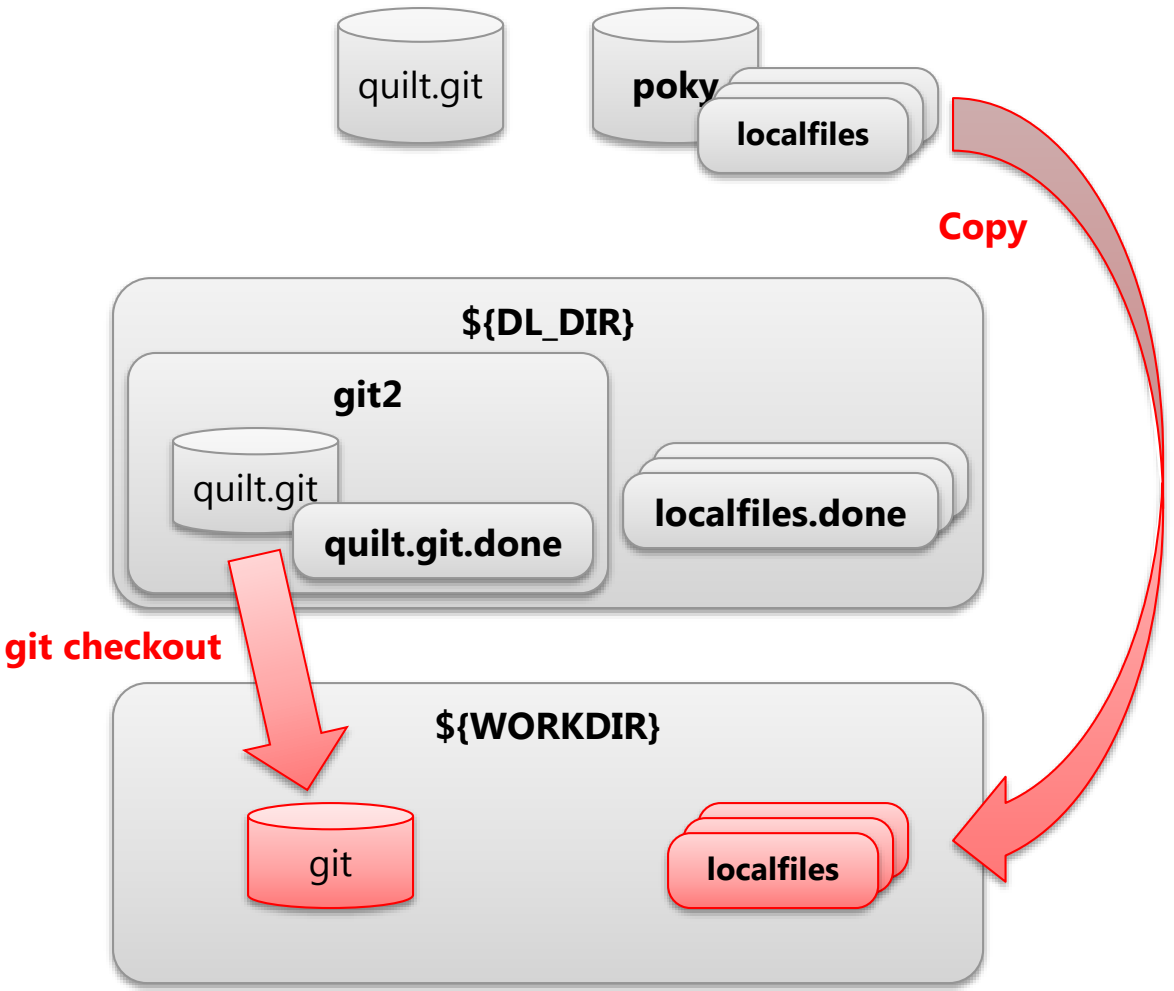
Build flow

bitbake tasks

(initialize)
do_fetch()
do_unpack()
do_debian_patch()
do_patch()
... (same as original)
...

bitbake variables

PN = quilt-native
PV = git${SRCPV}
PR = r0
DPN = quilt (${BPN})
DPR = 0



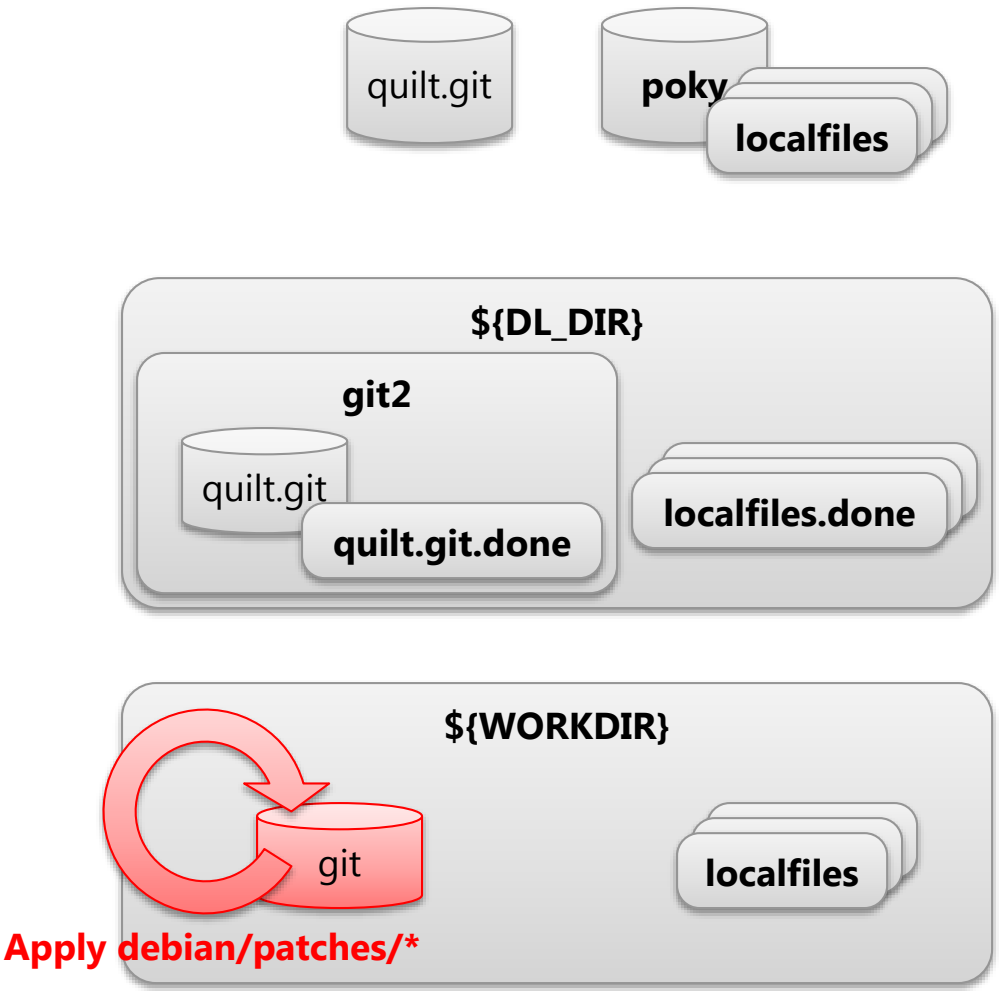
Build flow

bitbake tasks

(initialize)
do_fetch()
do_unpack()
do_debian_patch()
do_patch()
... (same as original)
...

bitbake variables

PN = quilt-native
PV = git${SRCPV}
PR = r0
DPN = quilt (${BPN})
DPR = 0



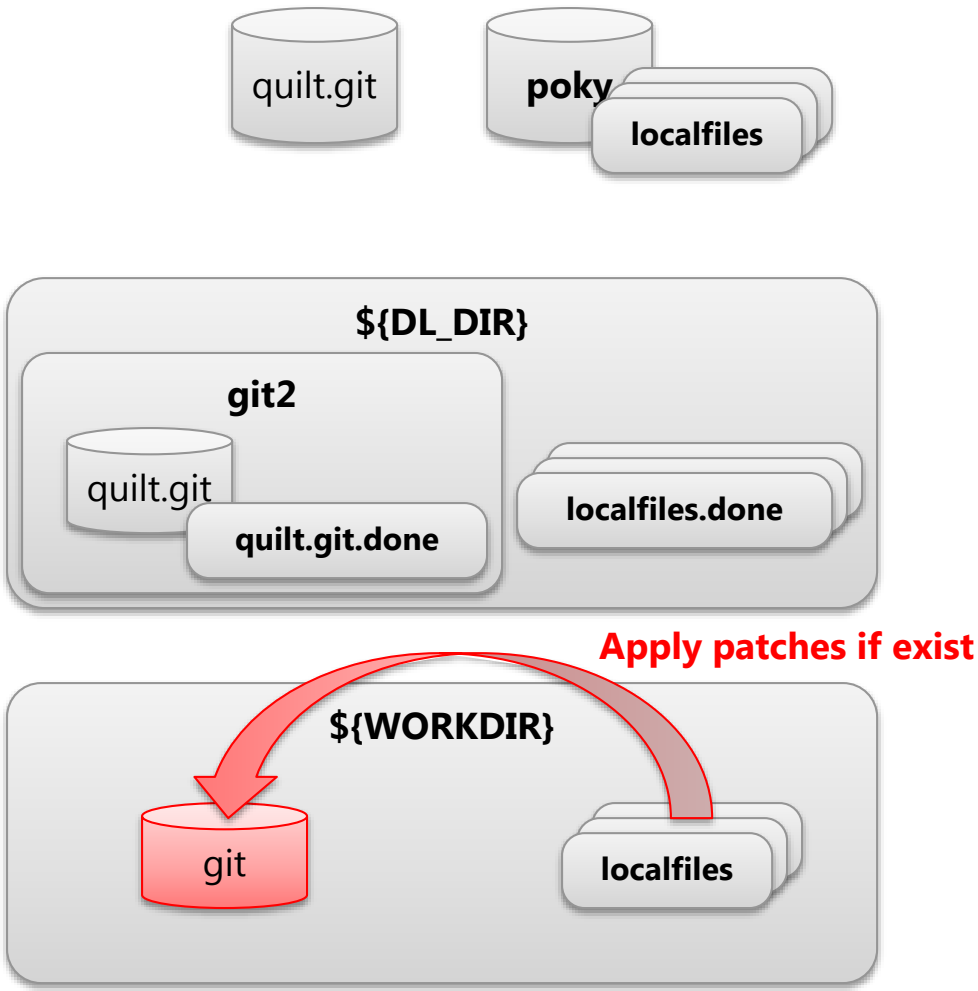
Build flow

bitbake tasks

(initialize)
do_fetch()
do_unpack()
do_debian_patch()
do_patch()
... (same as original)
...

bitbake variables

PN = quilt-native
PV = git${SRCPV}
PR = r0
DPN = quilt (${BPN})
DPR = 0



Meta-debian Quick Start

Preparation

- **Setup Poky and Meta-debian sources**

```
$ cd $WORKDIR
(checkout poky)
$ git clone git://git.yoctoproject.org/poky.git
$ cd poky
$ git checkout daisy

(checkout meta-debian)
$ git clone git://github.com/ystk/meta-debian.git
$ cd meta-debian
$ git checkout daisy
```

How to build recipes

- **Run startup script**

```
$ cd $WORKDIR
$ source /path/to/poky/oe-init-build-env
```

- **Add "meta-debian" layer to conf/bblayers.conf**

```
BBLAYERS ?= " ¥
  /path/to/poky/meta ¥
  /path/to/poky/meta-debian ¥
  "
```

- **Run bitbake**

```
$ bitbake <build_target>
```

Comparing Poky-debian and Meta-debian

	Poky-debian	Meta-debian
Poky version	Edison	Daisy
Debian version	6 (Squeeze)	8 (Jessie)
Kernel	LTSI + RT patch	LTSI + RT patch
Distribution Support	Debian 6	Debian 8
Status	Stable	Under development
Number of packages	440	70
Number of BSPs	10	1
Debian binary compatibility	NO	NO
Ability to use Yocto project tools	Absolutely not	??

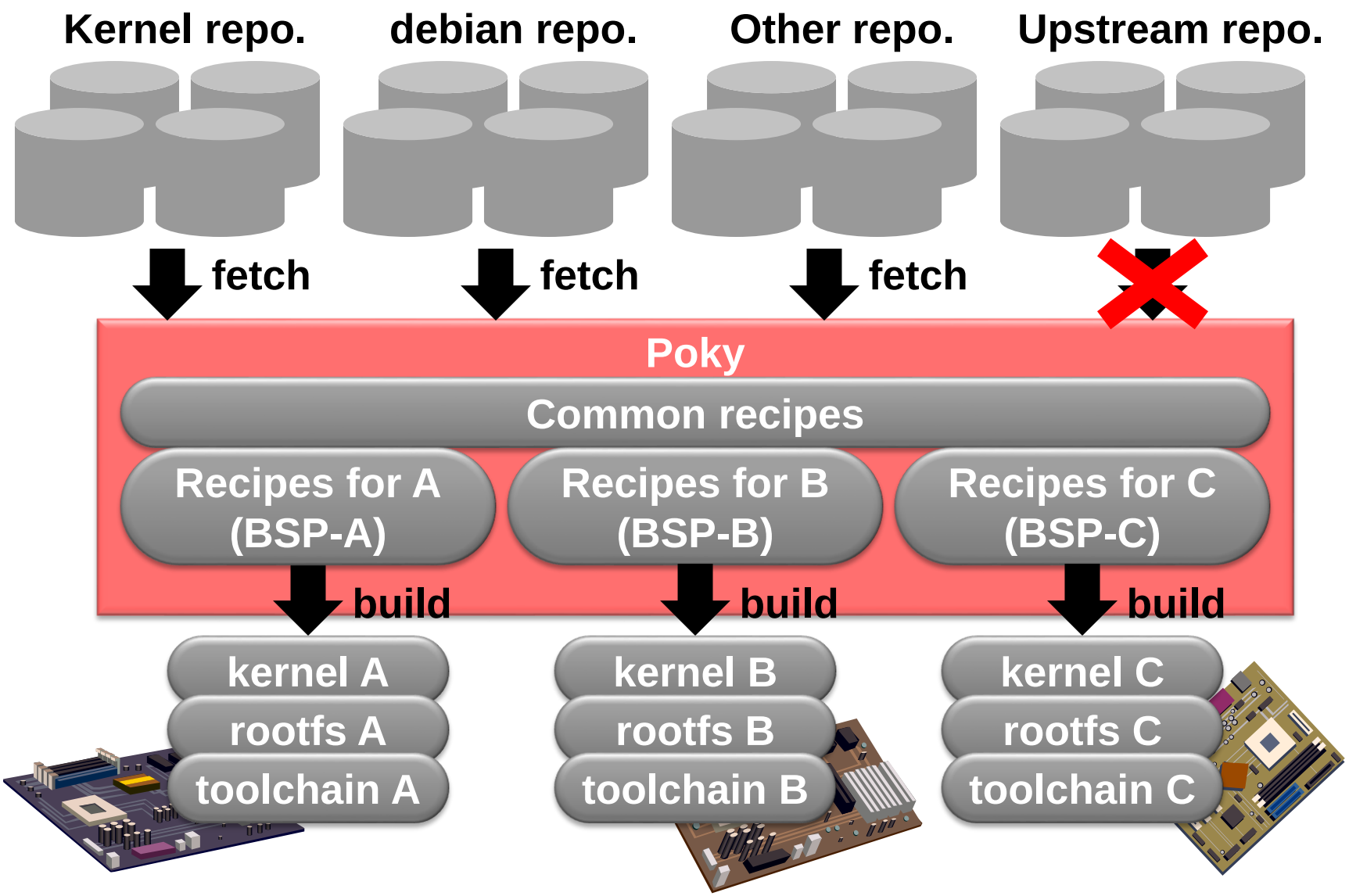
- **Please send your feedback**
  - E-mail: yoshitake.kobayashi@toshiba.co.jp

Thank you

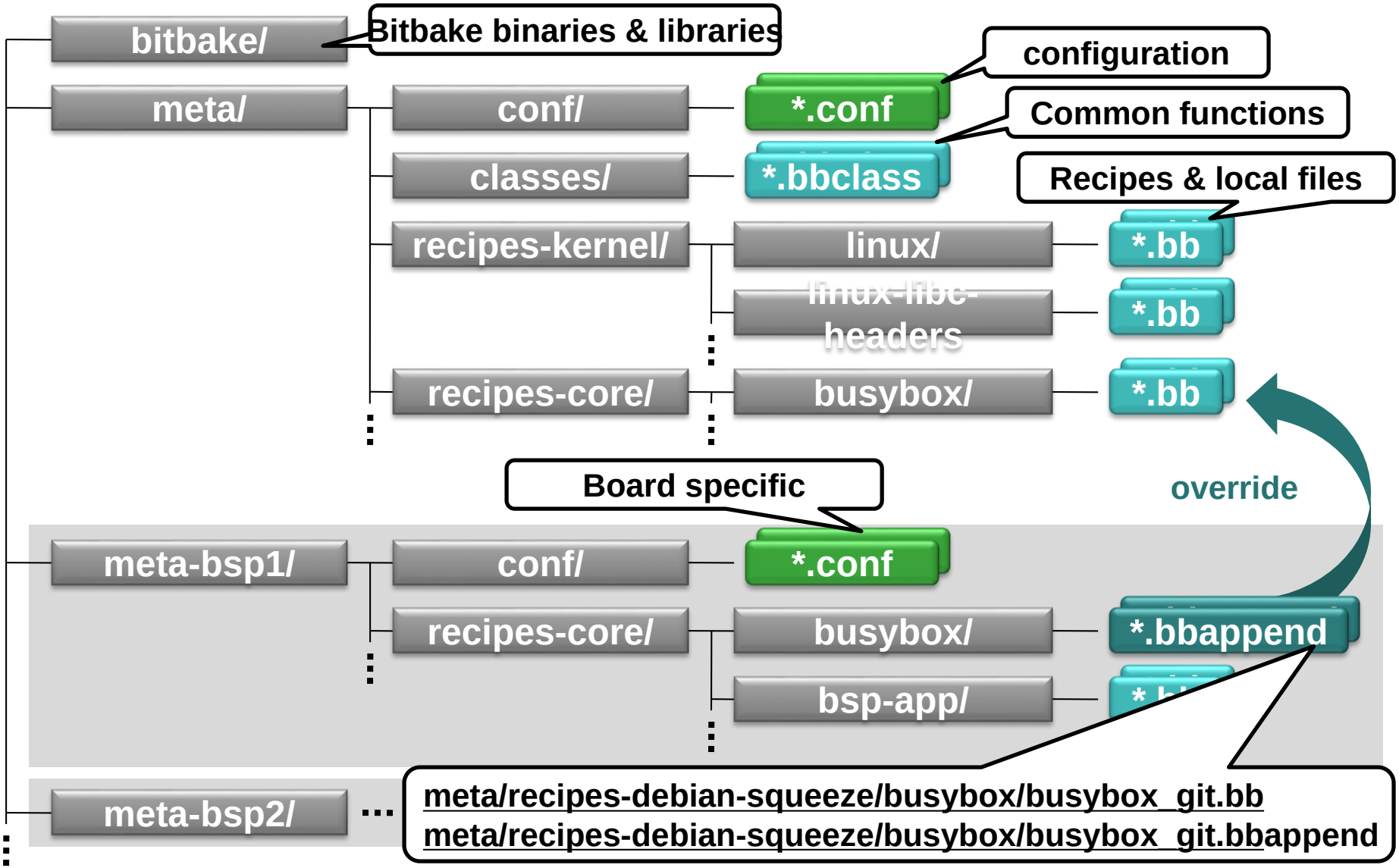
Appendix

More details for poky-debian

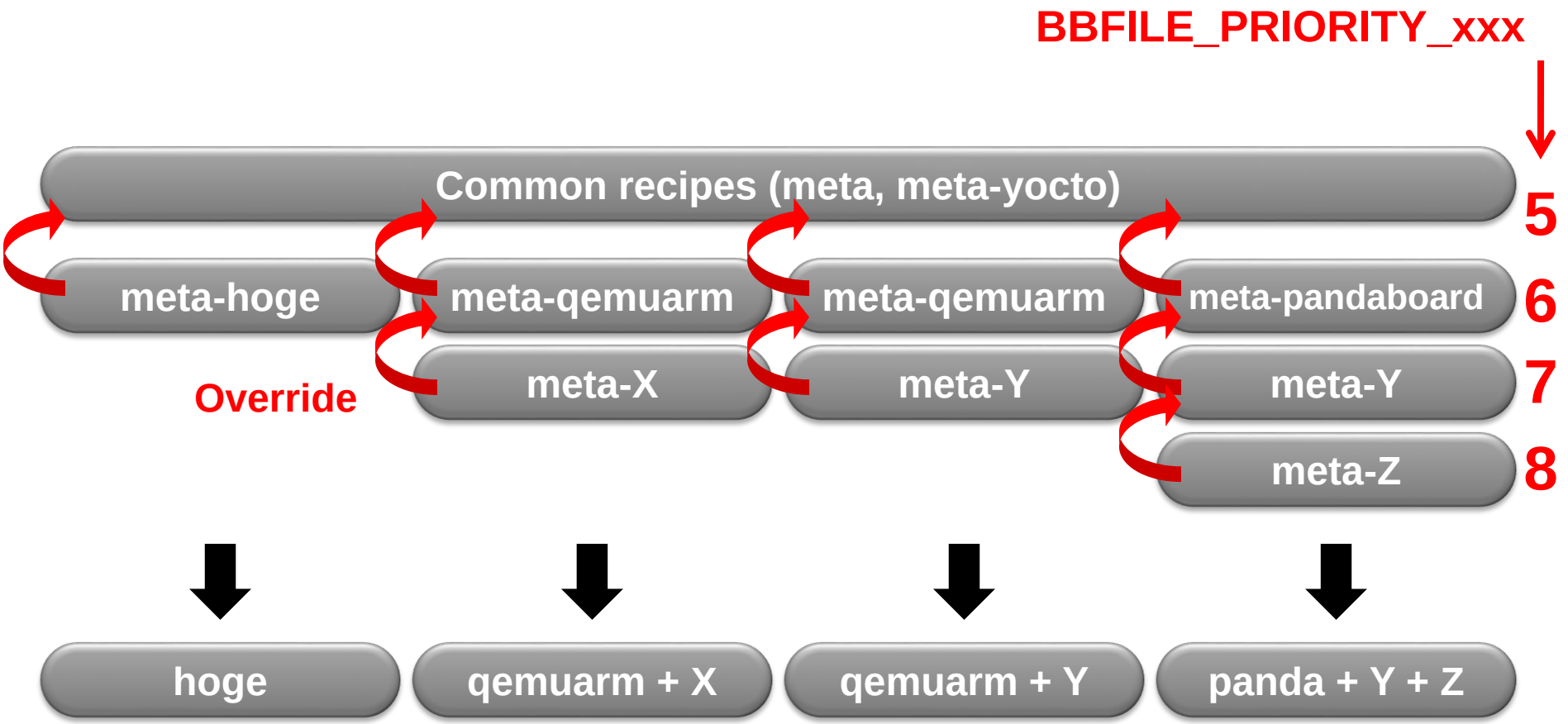
Poky-debian Overview



Directory structure



Layeis (meta-*)


BBFILE_PRIORITY_xxx
Common recipes (meta, meta-yocto)
5
meta-hoge
meta-qemuarm
meta-qemuarm
meta-pandaboard
6
Override
meta-X
meta-Y
meta-Y
7
meta-Z
8
hoge
qemuarm + X
qemuarm + Y
panda + Y + Z

Directory rules

- **recipes-debian-squeeze**
  - All recipes in this directory fetch all sources from “Git server”
    - DEBIAN_SQUEEZE_GIT_APP ?= git://github.com/ystk/debian-
  - All recipes need to “inherit debian-squeeze”
- **recipes-kernel**
  - All recipes in this directory fetch all sources from “Git server”
    - DEBIAN_SQUEEZE_GIT_KERNEL ?= "git://github.com/ystk/linux-poky-debian.git”
- **recipes-yocto**
  - All recipes in this directory fetch all sources from “Git server”
    - DEBIAN_SQUEEZE_GIT_YOCTO ?= git://github.com/ystk/tools-yocto1-“
    - These repositories are imported from non-Debian upstream used by original Poky without modification
  - All recipes need to “inherit debian-squeeze-yocto”
  - No need to be modified in BSP layers
- **recipes-poky**
  - All recipes in this directory never fetch source from remote servers (use only local file)
- **conf**
  - Layer specific configuration, machine configuration, etc.
  - All configuration files define variables applied ONLY to the layer
- **classes**
  - This directory includes only common “class files” inherited by recipes in recipes-*
  - No need to be modified in BSP layers

Contents

- **conf/layer.conf**
  - Layer specific configuration (See later for more details)
- **conf/bblayers.conf**
  - Define which layers are enabled

```
LCONF_VERSION = "4"

BBFILES ?= ""
BBLAYERS = " ¥
##COREBASE##/meta ¥
##COREBASE##/meta-yocto ¥
##COREBASE##/meta-beagleboard ¥
##COREBASE##/meta-target1/meta-sub1 ¥
##COREBASE##/meta-target1/meta-sub2 ¥
"
```

Don't modify

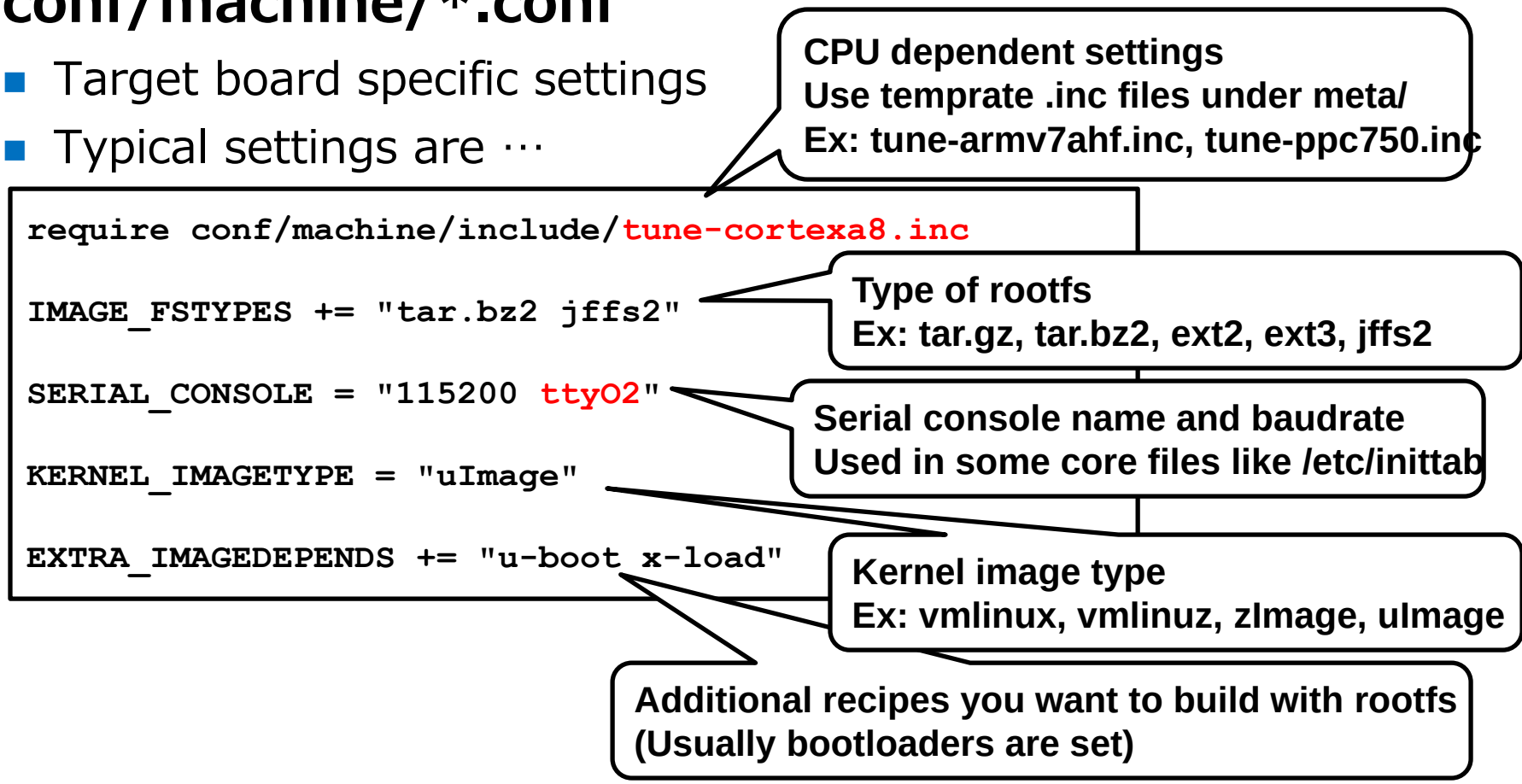
Don't modify (Always required)

Write additional layers

  - Automatically copied to the build directory by setup.sh
    - . ./poky-debian/setup.sh foo/bar meta-foo/meta-bar/conf/bblayers.conf → BUILD/conf/bblayers.conf

Contents

- **conf/machine/*.conf**
  - Target board specific settings
  - Typical settings are …



Contents

- **files/device-table/*.txt**
  - Device files which are installed statically
  - Add it to IMAGE_DEVICE_TABLES in core-image-*.bb
    - IMAGE_DEVICE_TABLES += “files/device-table/hoge.txt”
  - Typical lists are already defined in meta/files/device-table
    - Please define additional device files required only on target board

```
# path          type      mode    uid     gid     major   minor   start     inc     count
/dev              d        755      0       0         -       -       -       -         -
/dev/console      c        662      0       0         5       1       -       -         -
/dev/kmem         c        640      0      15         1       2       -       -         -
/dev/mem          c        640      0      15         1       1       -       -         -
/dev/null         c        666      0       0         1       3       -       -         -
/dev/tty          c        600      0       5         4       0       0       1         7
...
/dev/ttyO2        c        640      0       5       253       2       -       -         -
```

Board dependent device file

Contents

- **recipes-debian-squeeze/xxx/xxx.bb**
- **recipes-poky/xxx/xxx.bb**
- **recipes-kernel/xxx/xxx.bb**
  - Recipes which are required only in this layer
  - Please add a recipe under meta(common) if it's able to be shared with multiple layers

- **recipes-debian-squeeze/xxx/xxx.bbappend**
- **recipes-poky/xxx/xxx.bbappend**
- **recipes-kernel/xxx/xxx.bbappend**
  - All settings in xxx.bbappend are appended to xxx.bb
  - Ex: Override functions, add new functions, use another local file, etc.

layer.conf

- **Essential definitions**

```
BBPATH := "${BBPATH}:${LAYERDIR}"
BBFILES := " ${BBFILES} ¥
${LAYERDIR}/recipes-*/*/*.bb ¥
${LAYERDIR}/recipes-*/*/*.bbappend ¥
"
BBFILE_COLLECTIONS += "xxx"
BBFILE_PATTERN_xxx := "^${LAYERDIR}/"
BBFILE_PRIORITY_xxx = "8"
```

File paths

Layer name and priority

- **MACHINE**
  - Name of target machine
    - Ex: beagleboard, pandaboard, cubox
  - Machine independent layers should not define this value

layer.conf

- **Kernel information**
  - LINUX_REPO: Repository name (linux-poky-debian.git)
  - LINUX_SRCREV:
    Branch name (the latest commit is used) or Tag name or Commit ID
  - LINUX_CONF: Path to config (arch/arm/configs/hoge_defconfig)
  - See meta/classes/debian-squeeze-linux-checkout.bbclass for more details

- **RELEASE_VERSION**
  - Release version for each target
    - Output to /etc/debian_version (See debian-squeeze-files.bb for more details)
  - Format
    - ${debian_VERSION}-BOARDNAME-BOARDVERSION
    - Ex: 1.0.1-myboard-2.0

layer.conf

- **PREFERRED_PROVIDER_xxx = “x1”**
  - Bitbake uses “x1” as a ‘real’ recipe of “xxx”
  - Example:
    - PREFERRED_PROVIDER_gcc = “gcc-default”
    - PREFERRED_PROVIDER_gcc = “gcc-linaro”

- **PREFERRED_VERSION_xxx = “1.0”**
  - Bitbake uses “1.0” as a version of recipe “xxx”
    - Default version: git (the latest commit is used)
  - Example:
    - PREFERRED_VERSION_qt4-embedded = "4.8.2+dfsg-2debian1"

layer.conf

- **DISTRO_FEATURES**
- **DISTRO_FEATURES_append = “x11”**
  - http://www.yoctoproject.org/docs/1.1/poky-ref-manual/poky-ref-manual.html#ref-features-distro
  - Default value is defined in meta/conf/distro/include/default-distrovars.inc
  - It tells all recipes that specified “features” are enabled
  - Examples of features: ipv4, ipv6, x11, etc.

- **DEBIAN_SQUEEZE_FEATURES**
- **DEBIAN_SQUEEZE_FEATURES_append**
  - Deabin own features
  - Default value is defined in meta/conf/distro/debian-squeeze.conf

Update BSP

- **Update kernel**
  - Modify the following variables if you update kernel
    - LINUX_SRCREV, LINUX_CONF
- **Update userland files**
  - Modify recipes in recipes-debian/*, recipes-poky/*
- **Update release version**
  - First of all, build & install & run on the target system and confirm that there is no problem
  - Update RELEASE_VERSION and build image from nothing

Build images for release

- **Ex: Use meta-foo/meta-bar as target BSP**
  - NOTE: Please build from nothing (clean build)
    - Because all caches and unneeded sources should not be included in snapshot information
  - How to build

```
$ ls
poky-debian
$ . ./poky-debian/setup.sh foo/bar
$ bitbake core-image-base        ← for rootfs
$ bitbake meta-toolchain         ← for SDK
```

Recipe development for poky-debian

(Some slide has already obsoleted)

Recipe components

- **Each recipe consists of the following three file types:**
  - xyz_VERSION.bb
    - Core file. It includes or inherits other subfiles
  - xyz*.inc
    - Included by .bb or other .inc files
    - Usage of xyz.inc: `include xyz.inc`
  - Directories (files/, xyz/, xyz-VERSION/)
    - Includes local files (patchs, configuration files, etc.)

- **Class recipes**
  - `meta/classes/*.bbclass` has basic common functions which shared by all recipes that inherits it Take care to modify !
  - Usage of abc.bbclass: `inherit abc`

Package's own variables

- **PN**
  - PackageName (Ex: eglibc, busybox, etc.)
- **PV**
  - PackageVersion (Ex: 2.11.2-10, 1.17.1-8)
- **PR**
  - PackageRevesion (r0, r1, r2, ...)
- **WORKDIR**
  - Top of the build directory
  - build-$TARGET/tmp/work/$ARCH/${PN}-${PV}-${PV}
- **S**
  - Source code directory path used by do_unpack* and do_patch*
- **B**
  - Build directory path used by do_configure and do_compile
- **D**
  - Destination directory path used by do_install
- **These values are automatically set by bitbake**

How to write a recipe for package "hoge"

- **A. From scratch with a non-debian source (rare case)**
  - Add license information
  - Add SRC_URI
  - Set some variables or functions if needed

- **B. From scratch with a debian source (rare case)**
  - Add license information
  - Inherit debian class
  - Inherit autotools class if needed
  - Add SRC_URI if needed

- **C. Use an existing recipe with a debian source**
  - Copy poky's original recipe
  - Modify version
  - Modify license information
  - Inherit debian class

A-1. Add license information

```
LICENSE = "GPLv2"
LIC_FILES_CHKSUM = "file://COPYING;md5=1a2b3c..."
```

- **LICENSE: License name**
  - Usually written in ${S}/COPYING or ${S}/LICENSE
- **LIC_FILES_CHKSUM: License filename + its checksum**
  - Format: `file://FILENAME;md5=CHECKSUM`
  - Base path of FILENAME is "${S}"
  - You need to check the MD5 of COPYING by md5sum

A-2. Add "SRC_URI"

```
SRC_URI = " ¥
http://url.to.archive/foo.tar.gz ¥
file://default_config ¥
"
```

- **"SRC_URI" is a list of source codes, configurations, or other support files**
- **bitbake fetches, unpacks and patches all files listed in SRC_URI**
  - Archives (*.tar.gz, *.tar.bz2, etc.) are automatically unpacked
  - Patches (*.diff.gz, *.patch, etc.) are automatically patched

B-1. Add license information

```
LICENSE = "GPLv2"
LIC_FILES_CHKSUM = "file://COPYING;md5=1a2b3c..."
```

- Same as "A-1"

B-2. Inherit debian class

```
inherit debian-squeeze
```

- **Debian-squeeze class adds:**
  - do_fetch_srcpkg, do_unpack_srcpkg and do_patch_srcpkg
  - These functions fetch, unpack, patch essential sources automatically according to ${PN} and ${PV}
  - No need to write debian source URI in each package

B-3. Inherit autotools class

```
inherit autotools
```

- **autotools class adds:**
  - do_configure, do_make, do_install, etc. for autotools-based sources
  - Default functions are already defined in "base.bbclass", but some functions (do_configure, do_install, etc.) are empty
- **Don't inherit autotools if the source code is not based on autotools**
  - Please implement do_configure, do_compile, do_install, etc. from scratch

B-4. Add other required files to "SRC_URI"

```
SRC_URI = " ¥
file://COPYING ¥
file://default_config ¥
http://uri.to.sourcecode/rc-init.sh
"
```

- **Usually, only "inherit debian-squeeze" (source of package) is required**
- **Need to add file URIs to "SRC_URI" if the package requires some support files**
  - License file (COPYING)
  - Default configuration file for "menuconfig"
  - "rc" scripts to be installed to the target system
    - Ex: /etc/init.d/sshd

C-1. Copy poky's recipe from ORIGINAL

```
$ cp -r meta/ORIGINAL/recipes-???/foo meta/recipes-bar
```

- Check whether “foo/hoge_1.2.3.bb" is included in meta/recipes-debian-squeeze/ or not before copying

C-2. Modify version

```
$ cd meta/recipes-bar/hoe
$ mv hoge_1.2.3.bb hoge_1.2.1-4.bb
```

- **Replace Poky's version (1.2.3) by Debian's (1.2.1-4)**
- **The version of debian consists of two elements separated by "-"**
  - Format: UpstreamVersion-debianVersion
  - UpstreamVersion = 1.2.1
  - debianVersion = 4

C-3. Modify license information

```
LICENSE = "GPLv2"
LIC_FILES_CHKSUM = "file://COPYING;md5=1a2b3c..."
```

- Same as "A-1"
- Need to modify license information if Debian's license differs from poky's

C-4. Inherit debian-squeeze class

```
inherit debian-squeeze
```

- Same as "B-2"

C-5. Modification rule example

- Modifing a recipe according to the following rules

```
#
# ORIGINAL FILENAME
#

...
...
#...
...

#
# debian
#

OUR DEFINITIONS
```

Write the original filename not to forget it

Comment out only (Don't add / delete)

Write new definitions at the bottom

Example A: Recipe for "hello"

- Implement a recipe to build the following program

```
#include <stdio.h>

int main()
{
        printf("hello¥n");
        return 0;
}
```

```
default: clean hello
hello: hello.o
clean:
        rm -f hello *.o
```

Example A: Setup files

- **Make a directory for hello**

```
$ mkdir meta/recipes-test/hello
```

- **Make a source code directory and copy hello.c, Makefile and COPYING**

```
$ cd meta/recipes-test/hello
$ mkdir -p hello/src
$ cd hello/src
$ emacs hello.c
$ emacs Makefile
$ touch COPYING # dummy
```

- **Make a recipe file for hello**

```
$ emacs meta/recipes-test/hello/hello_1.0.bb
```

Example A: Write a recipe

- Add license information
- Add source files ("src") to "SRC_URI"
- Define "S" and "B" for hello
- Define do_install function to install binaries generated by our Makefile

- Run "bitbake hello"

Example A: Answer

```
LICENSE = "tmp"
LIC_FILES_CHKSUM = ¥
"file://COPYING;md5=d41d8cd98f00b204e9800998ecf8427e"

SRC_URI = "file://src"

S = ${WORKDIR}/src
B = ${S}

do_install() {
        install -d ${D}/${bindir}
        install -m 0755 ${B}/hello ${D}/${bindir}
}
```

Example C: Recipe for "sed"

- **Copy a recipe from ORIGINAL**

```
$ cp meta/ORIGINAL/recipes-extended/sed meta/recipes-test
```

- **Remove un-required files**

```
$ rm -f sed_4.1.2.bb sed-4.1.2/
```

- **Rename "4.2.1" to the debian version**
  - Check the version of source archive for Debian

- **Inherit debian-squeeze class**
- **Comment out all "SRC_URI"-related lines**

Example C: Recipe for "sed"

```
#
# sed_4.2.1.bb
#

DESCRIPTION = "sed is a Stream EDitor."
HOMEPAGE = "http://www.gnu.org/software/sed/"
...

#SRC_URI = "${GNU_MIRROR}/sed/sed-${PV}.tar.gz"

#SRC_URI[md5sum] = "f0fd4d7da574d4707e442285fd2d3b86"
#SRC_URI[sha256sum] =
"8773541ce097fdc4c5b9e7da12a82dffbb30cd91f7bc169f52f05f93b7fc3060"

inherit autotools update-alternatives gettext

...

BBCLASSEXTEND = "native"

#
# debian
#

inherit debian-squeeze
```

Kernel build

debian kernel repositories

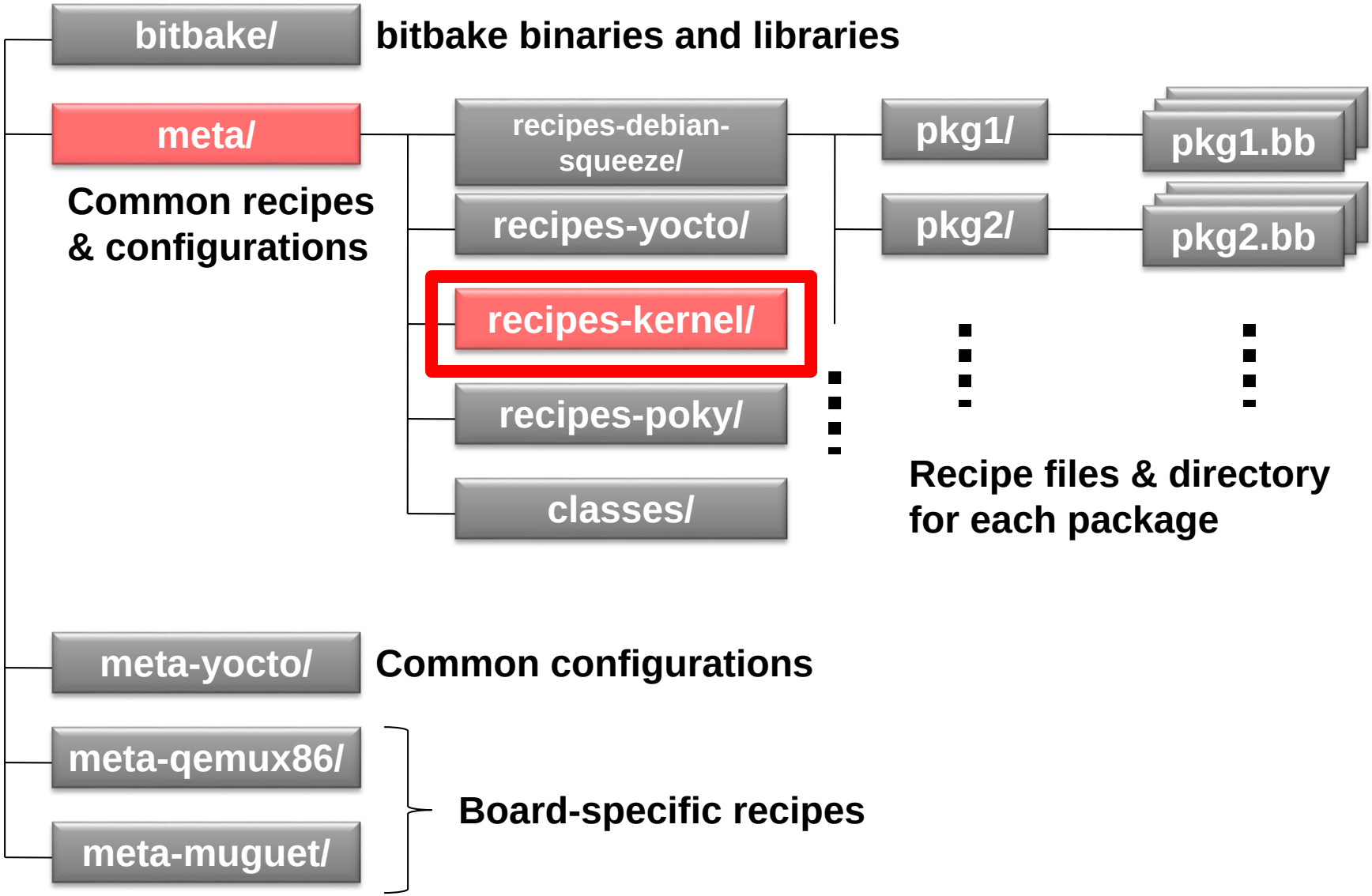
- **Sample kernel sources are able to download by the following command**

```
git clone git://github.com/ystk/linux-poky-debian.git
```

- **The kernel repository consists:**
  - Source code branch
    - v3.0-rt, etc.
    - Modifications for the target board

- **Poky-debian estimate to fetch all kernel source from git reposirtory via git protocol**
  - Need to prepare sources and configurations using SDK (toolchain) of poky-debian before build

Kernel recipes in poky-debian



Kernel recipes in poky-debian

- **meta/recipes-kernel/linux/linux_git.bb**
  - A sample recipe to build generic linux kernel
  - Fetch linux kernel source from git server
  - Inherit "debian-squeeze-linux.bbclass" instead of "debian-squeeze.bbclass"

- **meta/recipes-kernel/linux/linux-libc-headers_git.bb**
  - Include kernel headers (Ex: /usr/include/linux/*.h)
  - Used to build other userland packages
  - Fetch the same file as "linux_git.bb"
  - Inherit "debian-squeeze-linux-libc-headers.bbclass“ instead of "debian-squeeze.bbclass"

linux_git.bb

- **Variables**
  - LINUX_REPO
    - Name of kernel repository (Ex: linux-poky-debian.git)
    - Searched from ${DEBIAN_SQUEEZE_GIT}
  - LINUX_BRANCH_SRC
    - Name of source code branch
    - Default: "master"
  - LINUX_BRANCH_CONF
    - Name of configuration branch
    - Default: "configs"
  - LINUX_COMMIT_SRC / LINUX_COMMIT_CONF
    - Commit ID (hash value) of source code / configuration
  - LINUX_CONF
    - Name of configuration file in "configs" directory

linux_git.bb

- **Example 1**

```
LINUX_REPO = "linux-poky-debian.git"
LINUX_BRANCH_SRC = "v3.0-rt"
LINUX_BRANCH_CONF = "configs"
LINUX_CONF = "configs/v3.0/arm/versatile_defconfig"
```

  - Fetch github.com/ystk/linux-debian.git
  - Use the newest commit of "v3.0-rt" branch as a source
  - Use "versatile_defconfig" in configs branch as a configuration

- **Example 2**

```
LINUX_REPO = "linux-debian.git"
LINUX_COMMIT_SRC = "eb25ca22426dbaea10a4748c8741ccbc3aaa24c8"
LINUX_COMMIT_CONF = "1ea6b8f48918282bdca0b32a34095504ee65bab5"
LINUX_CONF = "configs/v3.0/arm/versatile_defconfig"
```

  - Use two commits ("`eb2...`" and "`1ea...`") as a source & conf

How to build?

- **bitbake kernel**

```
$ bitbake virtual/kernel
```

- **Build directory**
  - build-${MACHINE}/tmp/work/${MACHINE}-poky-linux/linux-git-r0

- **Deployment**
  - build-${MACHINE}/tmp/deploy/images/zImage-${MACHINE}.bin

How to customize the kernel? (menuconfig)

- Launch "screen"
- You can modify the configuration you choose

```
$ bitbake -c clean virtual/kernel
$ rm -f sstate-cache/sstate-linux-*

$ bitbake -c menuconfig virtual/kernel
```

- You will get the following message:

```
WARNING: Screen started. Please connect in another terminal with
"screen -r devshell"
```

- Generate new screen and type:

```
$ screen -r devshell
```

- Retry bitbake

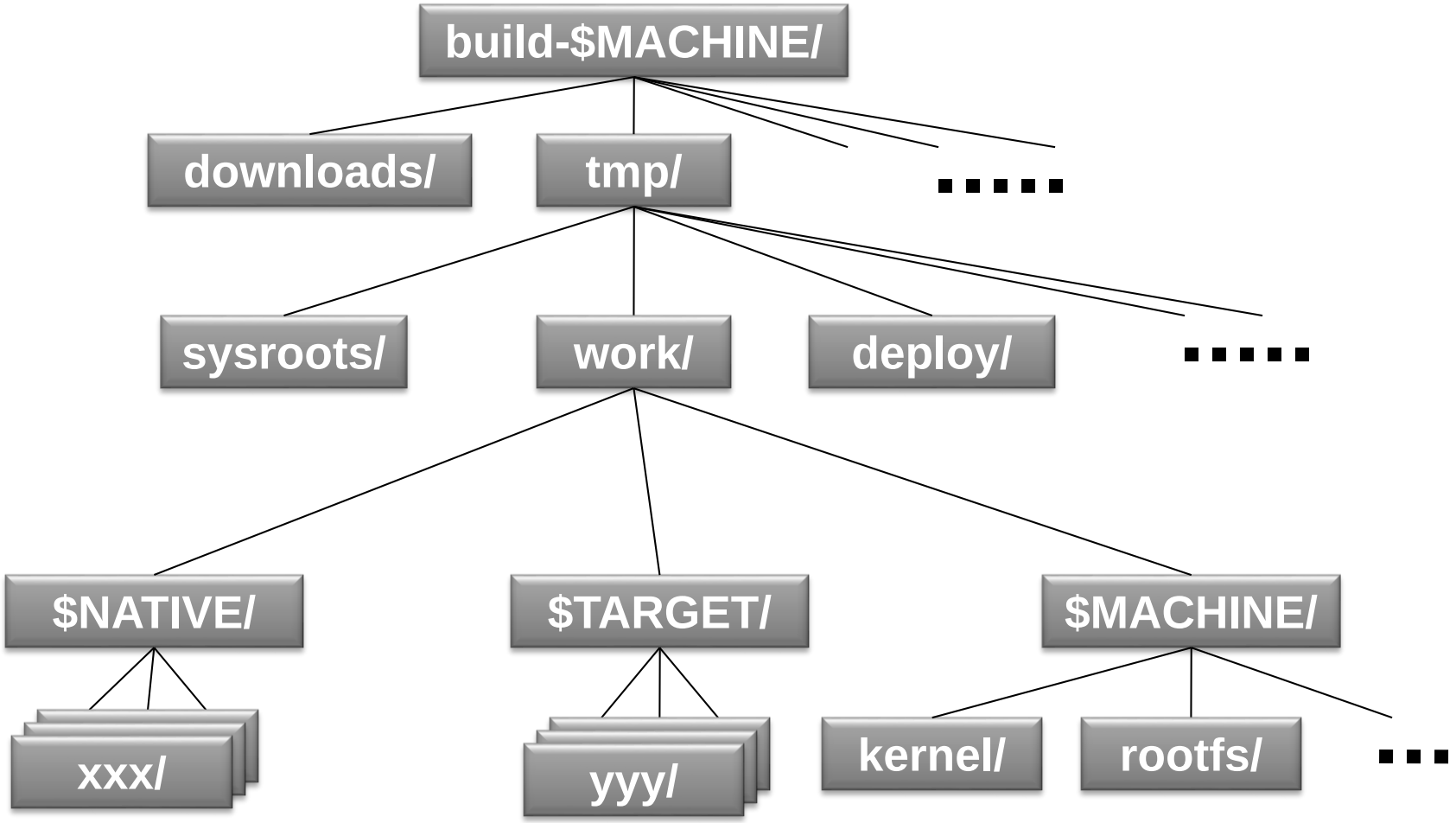
Debug

Build directory and deployment

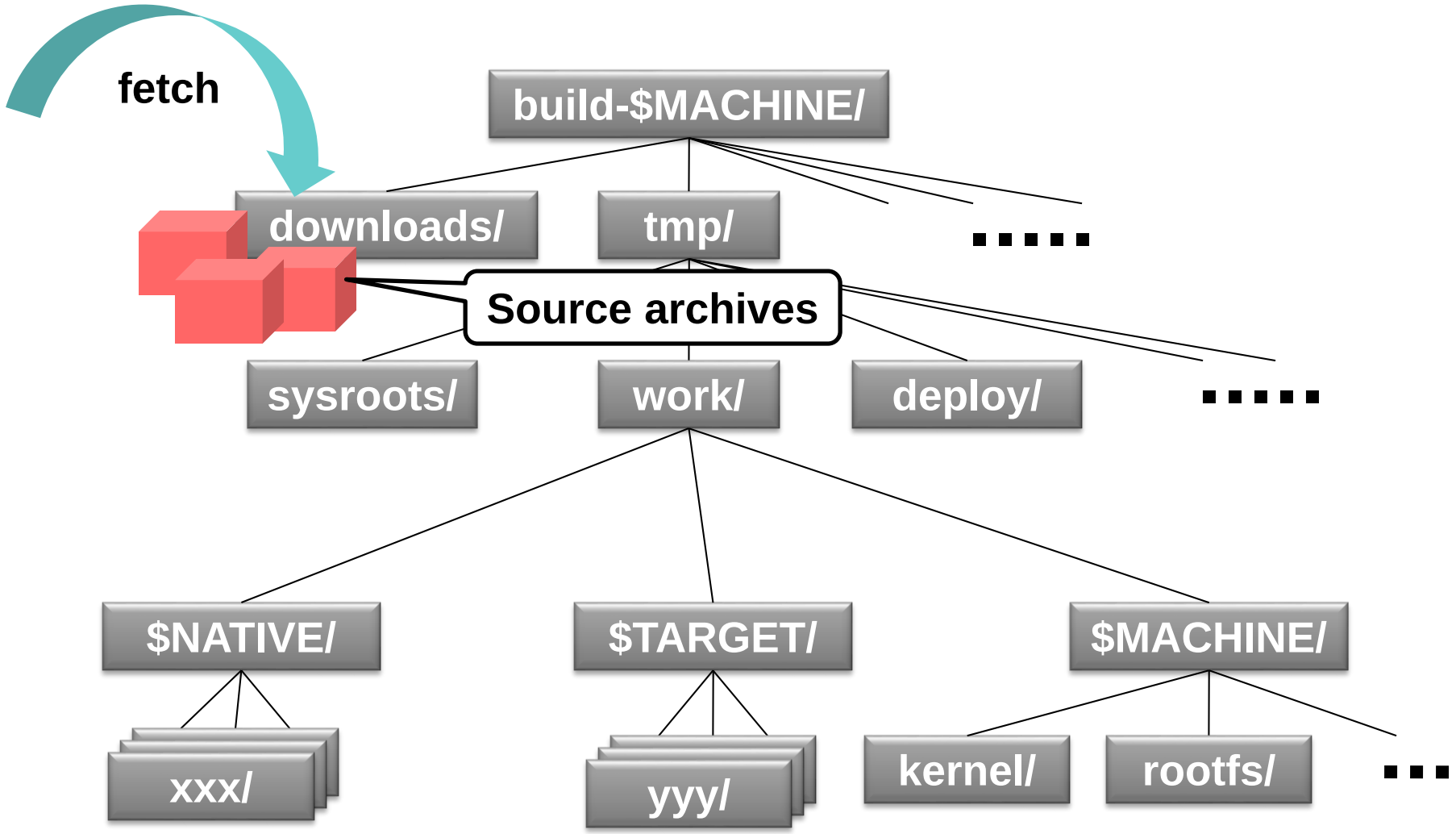
- **setup.source generates a build directory which named "build-$MACHINE"**
  - bitbake outputs all files only to this directory

- **The build directory consists of...**
  - conf
    - Includes configuration files copied by setup.source
  - downloads
    - Includes all downloaded files fetched by do_fetch*
  - tmp/work
    - Work directories that bitbake uses to build each recipe
  - tmp/sysroots
    - Includes tools & libraries shared by all recipes
    - Ex: native tools, cross compiler, cross libraries, headers, etc.
  - tmp/deploy
    - Final outputs are put into this directory
    - kernel, rootfs, toolchain, etc.

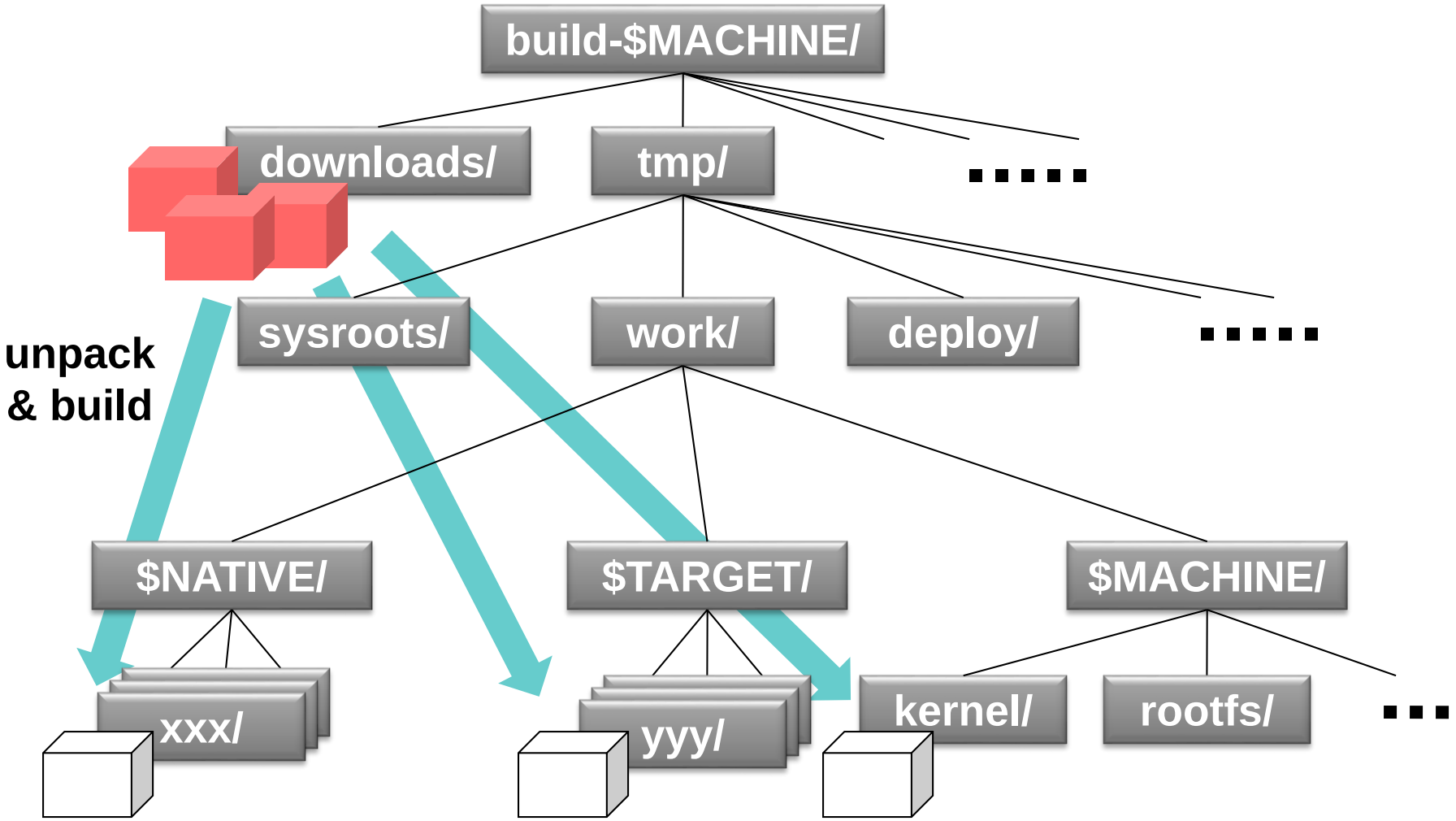
Build directory and deployment



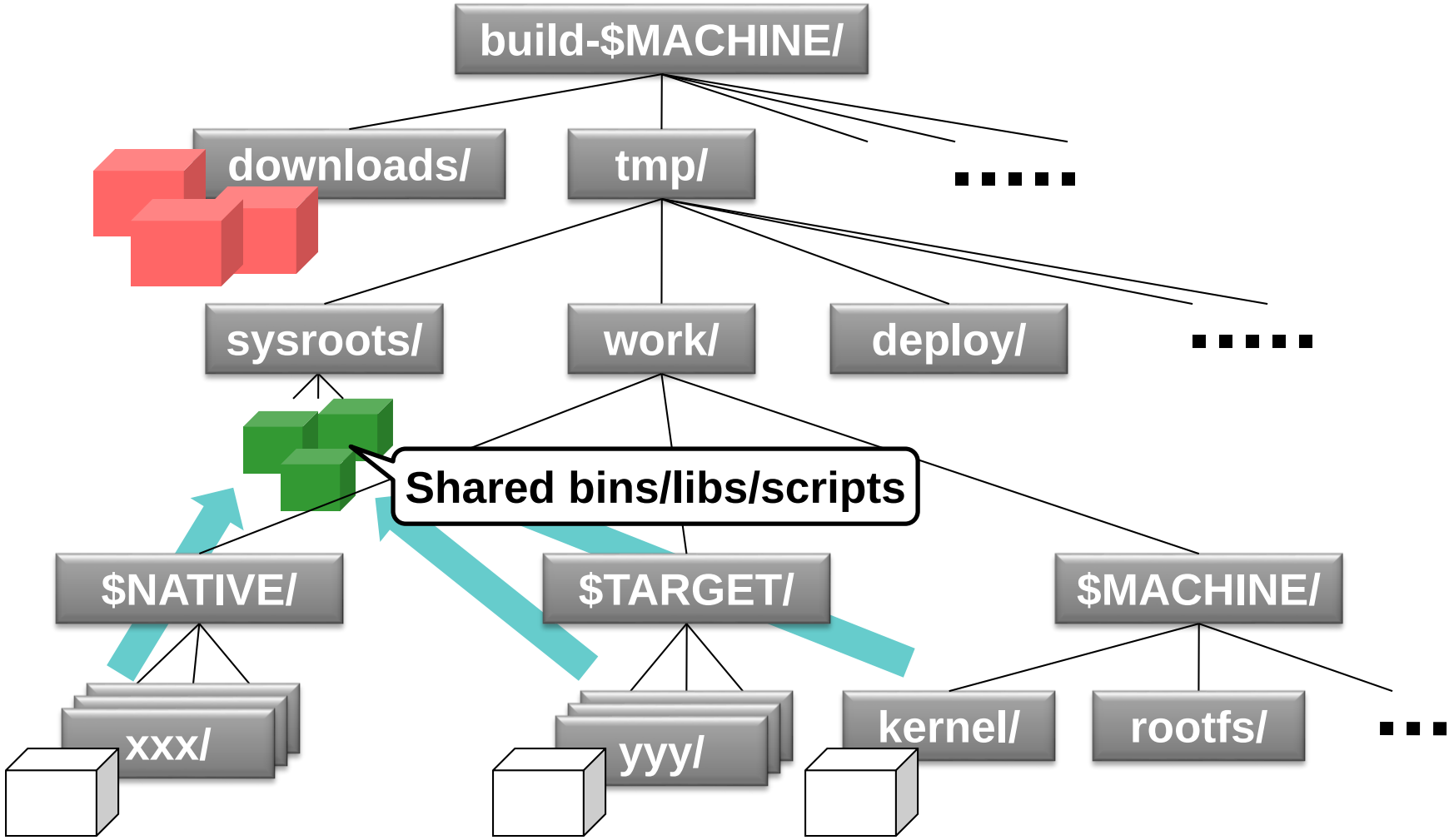
Build directory and deployment



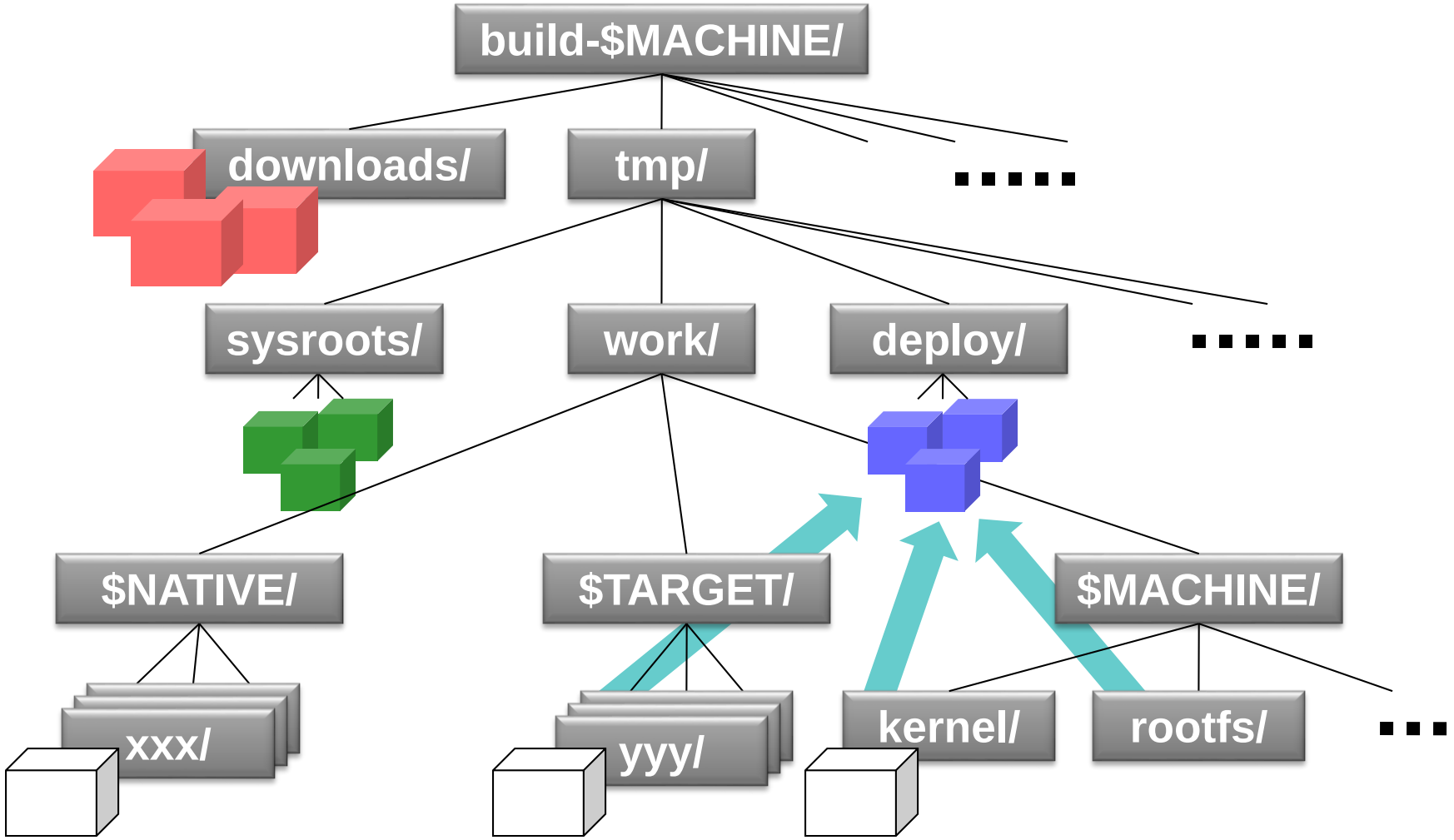
Build directory and deployment



Build directory and deployment



Build directory and deployment



Build directory and deployment

- **Each package build directory consists...**
  - temp
    - Includes all log files and 'real' build scripts generated by bitbake according to recipes
    - Please check logs under this directory when some errors occur
  - ${S} = ${PN}-${PV}/
    - Source directory unpacked by do_unpack
  - ${D}: image/
    - All outputs are put into this directory (usually by "make install")
  - deploy-debs
    - All packaged files named "*.deb" are put into this directory

Errors

- Fetch errors caused by repository servers
- Errors caused by source directory structure
- Path-related errors
- libtool-related errors
- Manual-related errors
- Patch failures when we apply some poky's patches

Fetch errors caused by repository servers

- **Source archives not found**
  - Some packages don't exist on repository servers
  - Please add the package to debian git repository yourself

- **debian package name is different from poky's**
  - Ex: debian="lm-sensors-3" poky="lm-sensors"
  - Use “DEBIAN_SQUEEZE_SRCPKG_NAME"

```
inherit debian
debian_SRCPKG_NAME = "lm-sensors-3"
```

lm-sensors_xxx.bb

Errors caused by source directory structure

- **Structure of some source directories are different from poky's source directories**
  - debian
    pkgname-0.1-2/
      Makefile, src.c, configure, ...
  - poky
    pkgname-0.1/
      srcdir, support-tools, ...
        └→ Makefile, src.c, ...

- **Please fix "S" and related functions such as do_unpack**

Path-related errors

- **Sometimes, Makefile is hard-corded**
  - Ex: CC = gcc
  - Cannot use cross compiler in poky in this case
- **You may have some patches that resolve this problem**
  - Ex: fix-path-xxx.patch
- **You need to fix paths if there is no patches in poky**

libtool-related errors

- **Some packages fail with libtool-related errors**
- **libtool**
  - One of autotools
  - Compile, build, install libraries
- **poky-debian uses internal libtool to build each package**
  - Some packages need to be patched to use internal libtool
  - Please search "libtool-xxx.patch" and apply it when you get some libtool-related errors

Manual-related errors

- **Some packages fail with manual-related errors**
  - manpage, documents, etc.
  - Ex: help2man
- **Please fix Makefile**
  - Delete rules to build manual or documents in Makefile
  - No need to build manual or documents because we are generating root file systems not for generic PC but for embedded systems

Patch failures

- **Need to apply some patches included in poky-debian to build packages**
- **Usually, some patches reject ☹**
  - Because the pacthes assume that it is applied to poky's source
  - But we use debian's source now
- **Please fix patches by yourself..**

How to create a new patch?

- **First, backup (1) the original source files (such as Makefile)**
- **Second, do bitbake**
  - The error occurs
- **Third, (2) modify the source files and retry bitbake**
- **Create a patch from difference between (1) and (2) using "diff" command**
- **Add your patch to SRC_URI**

```
SRC_URI = "my-patch.patch"
```

Packaging

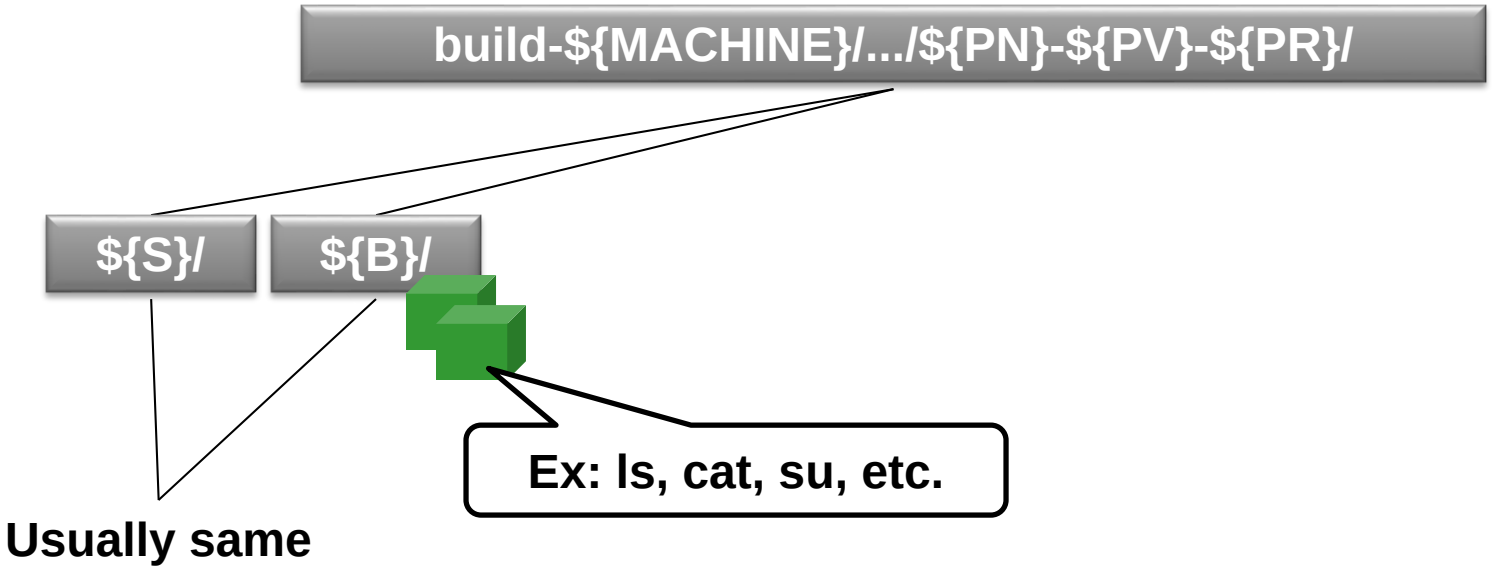
Build directory and deployment

- **Build directory of each package consist of...**
  - temp
    - Includes all log files and 'real' build scripts generated by bitbake according to recipes
    - Please check logs under this directory when some errors occur
  - ${S} = ${PN}-${PV}/
    - Source directory unpacked by do_unpack
  - ${D}: image/
    - All outputs are put into this directory (usually by "make install")
  - deploy-debs
    - All packaged files named "*.deb" are put into this directory

***.deb is a binary package format of Debian**

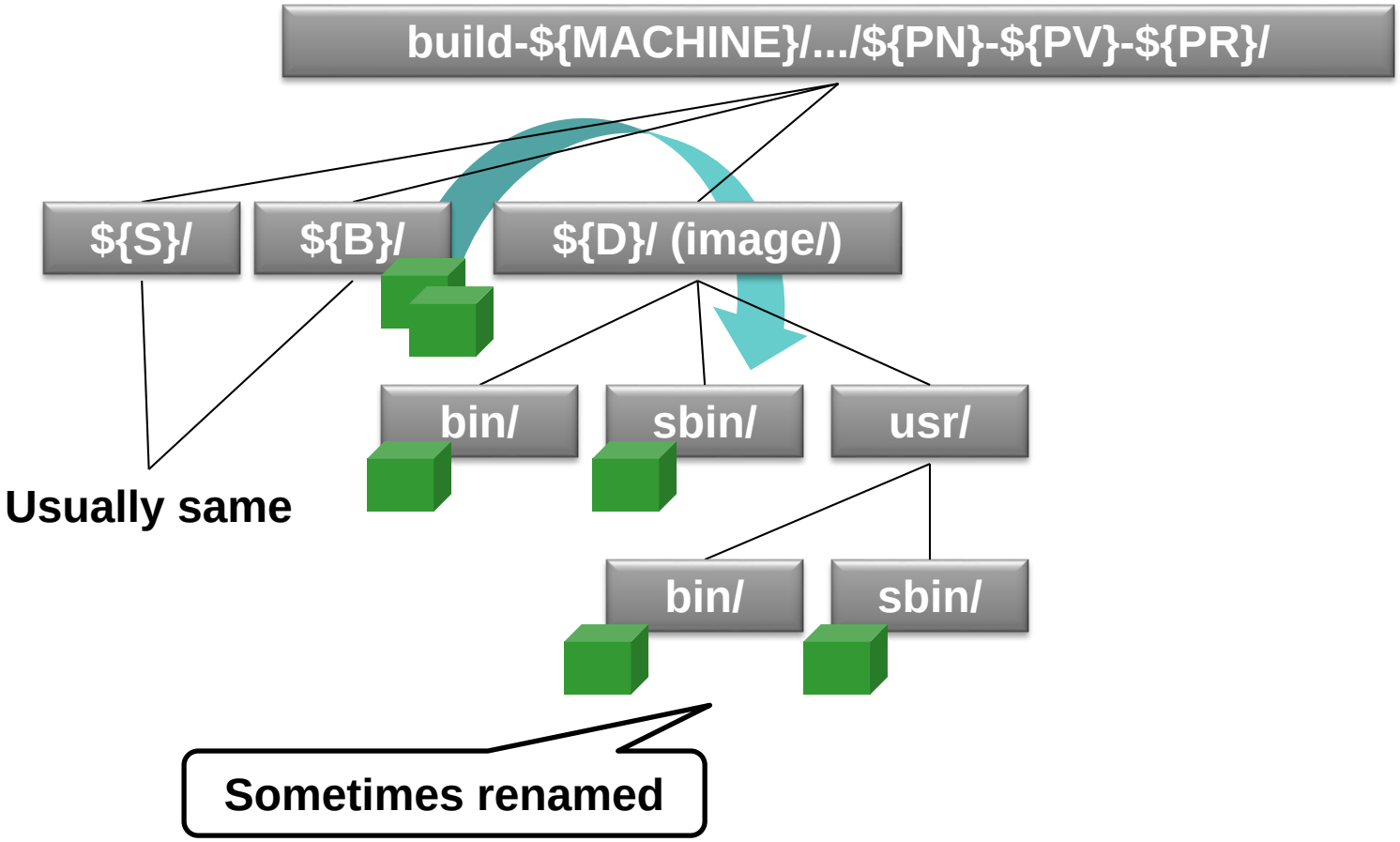
How *.deb packages are created?

- After do_compile, we have some outputs of the package under ${B}



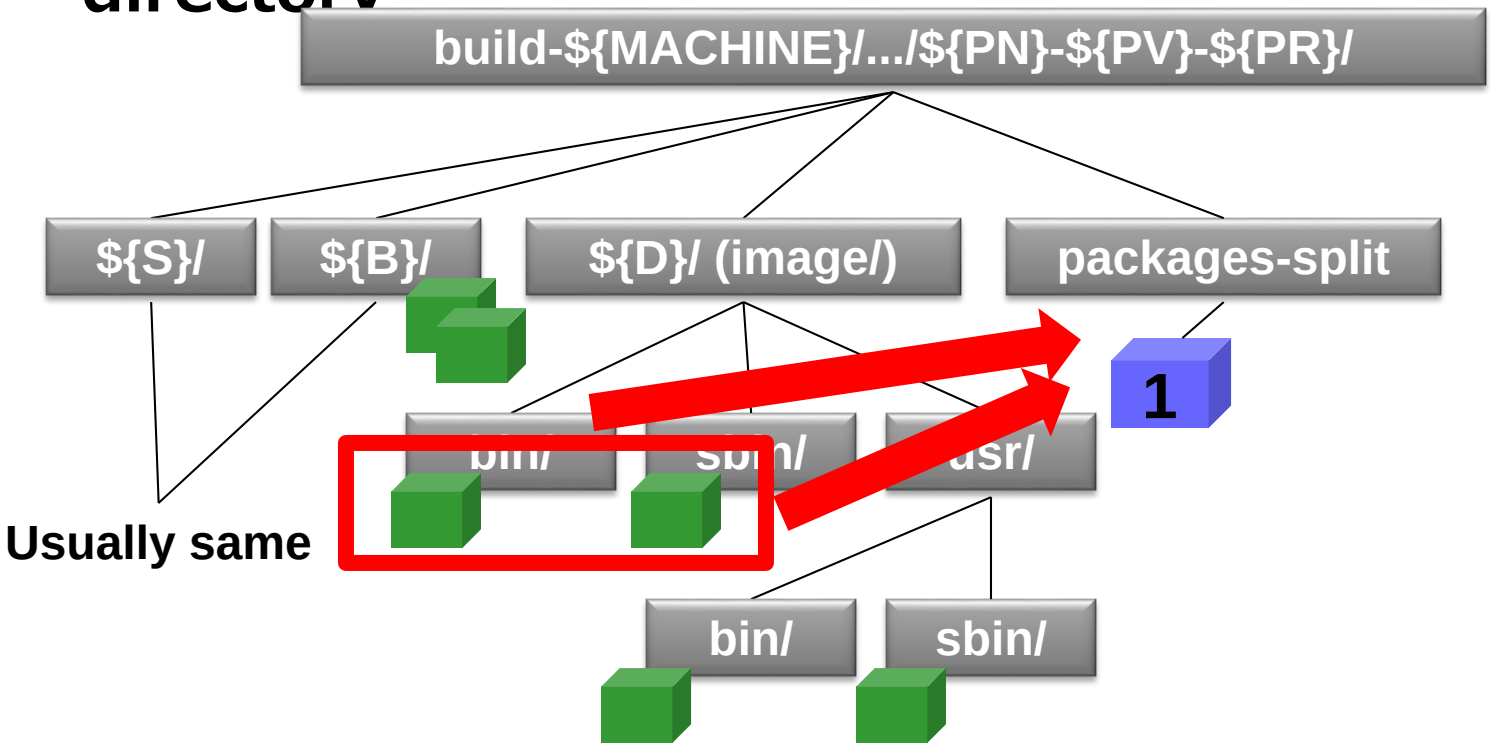
How *.deb packages are created?

- After do_install, outputs are put into ${D} according to "make install"



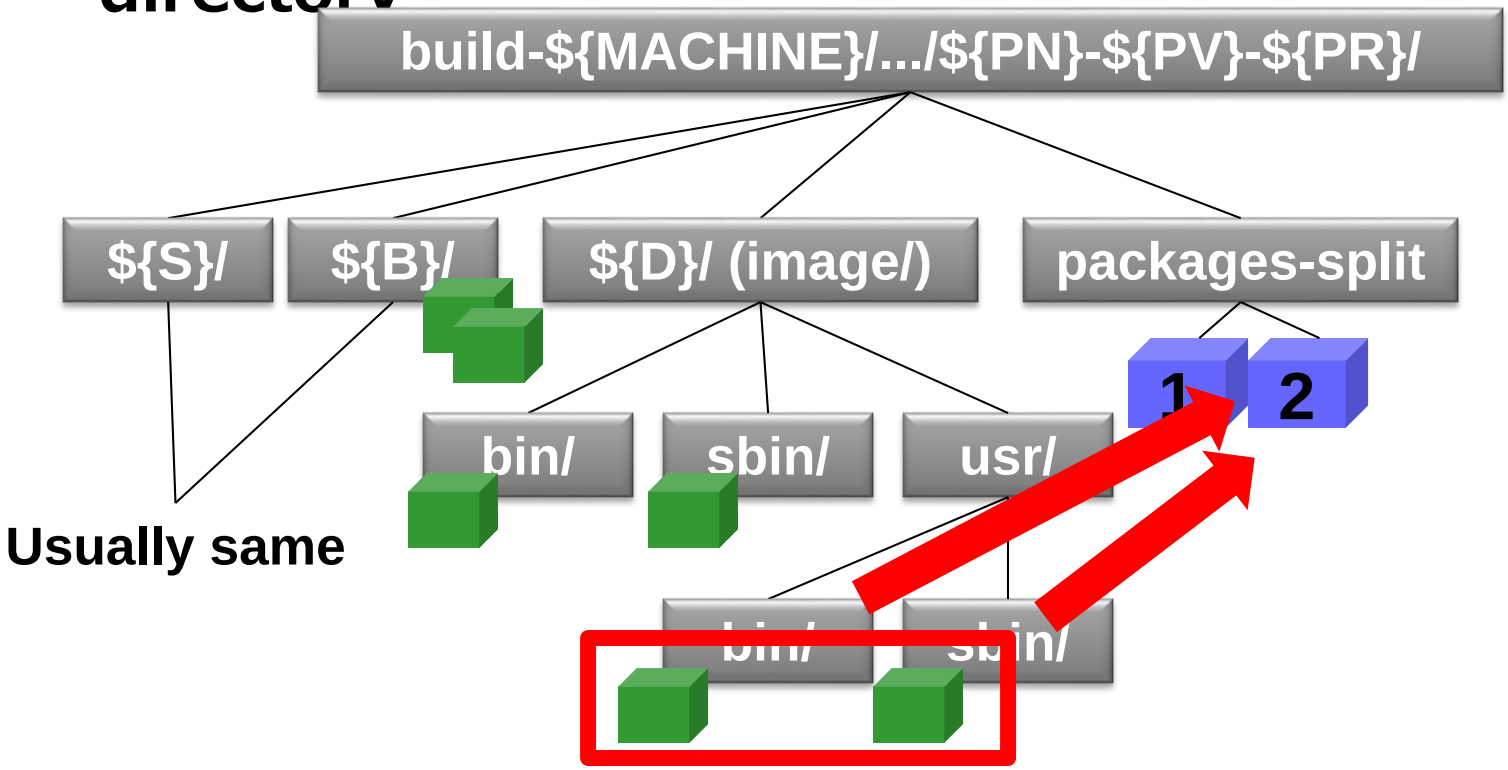
How *.deb packages are created?

- In do_package* functions, each output file are installed to "packages" and "packages-split" directory



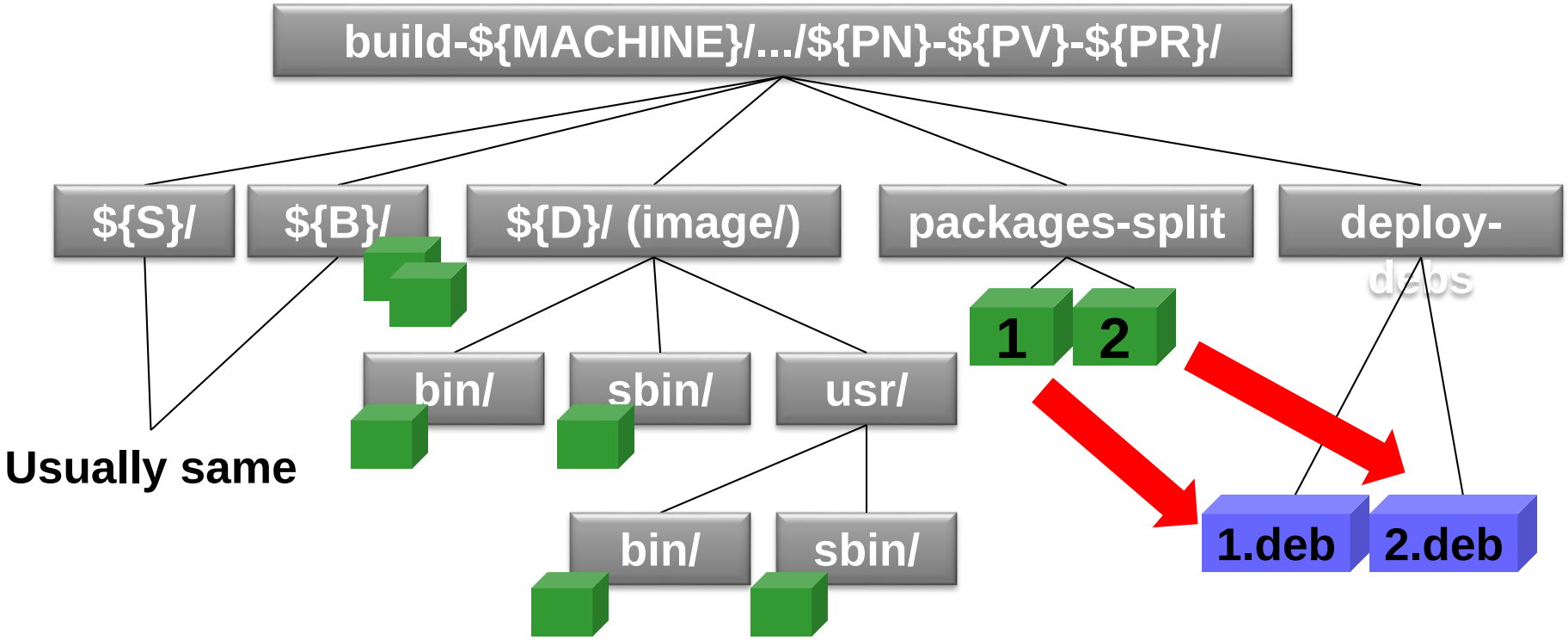
How *.deb packages are created?

- In do_package* functions, each output file are installed to "packages" and "packages-split" directory



How *.deb packages are created?

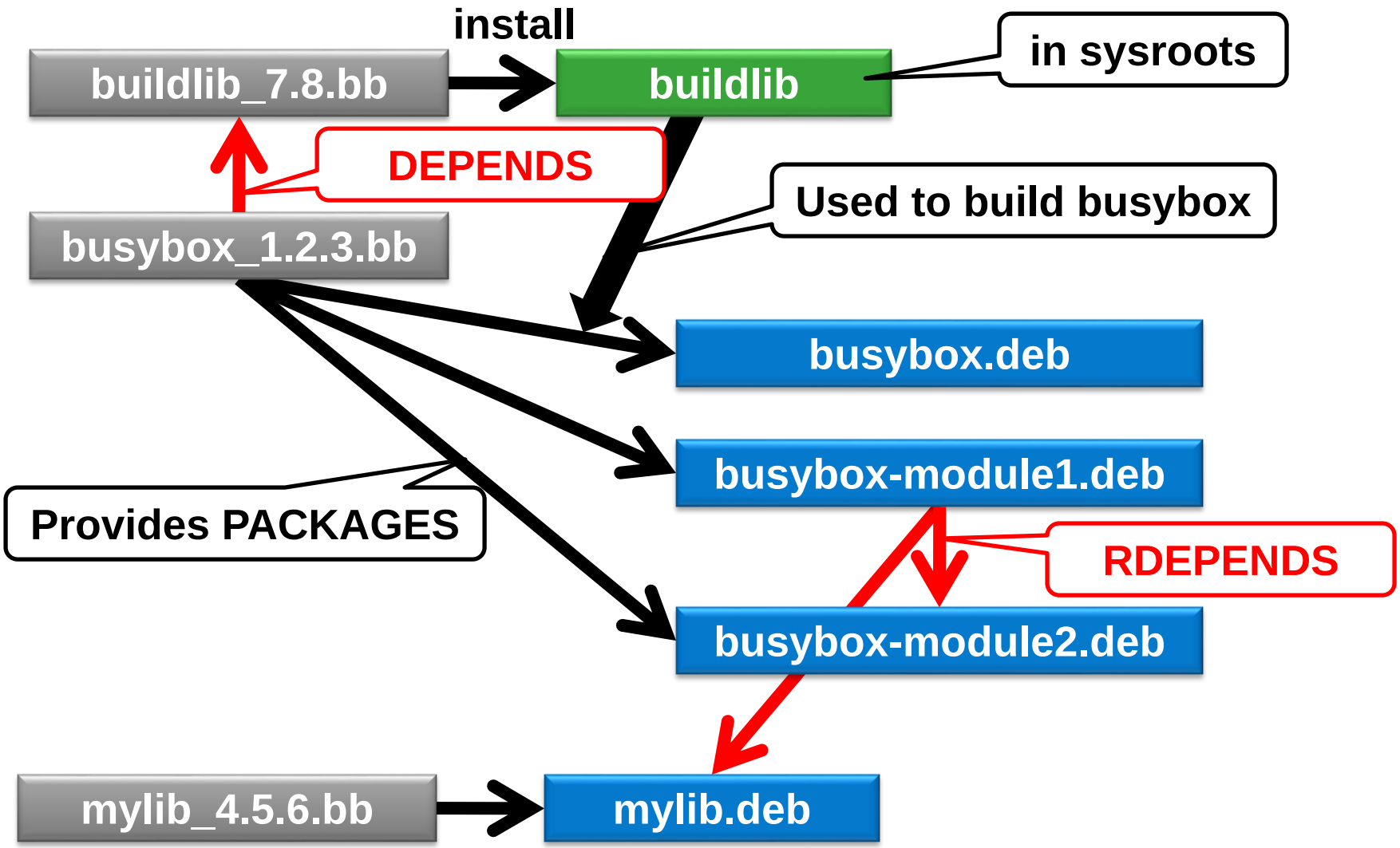
- Finally, split *.debs are generated in "deploy-debs"



How to control packaging?

- **We can use the following variables to control packaging**
  - PACKAGES
    - The list of package name
    - Ex: `PACKAGES = "busybox busybox-module1 busybox-module2"`
  - FILES_xxx
    - The list of files included in package "xxx"
    - Ex: `FILES_busybox-module1 = "${bindir}/bin1 ${sbindir}/bin2"`
  - RDEPENDS_xxx
    - The list of packages "xxx" depends on
    - Ex: `RDEPENDS_busybox-module1 = "busybox-module2 mylib"`
    - NOTE: "DEPENDS" != "RDEPENDS"
      - DEPENDS: "Build-time" dependency
      - RDEPENDS: "Runtime" dependency

Dependency


install
buildlib_7.8.bb
buildlib
in sysroots
DEPENDS
Used to build busybox
busybox_1.2.3.bb
busybox.deb
busybox-module1.deb
Provides PACKAGES
RDEPENDS
busybox-module2.deb
mylib_4.5.6.bb
mylib.deb